Software Engineering Institute

Proceedings of the International Workshop on the Foundations of Service-Oriented Architecture (FSOA 2007)

Grace A. Lewis
Dennis B. Smith

June 2008

SPECIAL REPORT
CMU/SEI-2008-SR-011

Research, Technology, and System Solutions Program

Carnegie Mellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (http://www.sei.cmu.edu/publications/pubweb.html).

Table of Contents

List of Figures

List of Tables

Acknowledgments

Our thanks to Carnegie Mellon® Software Engineering Institute (SEI) for sponsoring this Independent Research and Development project in fiscal year 2007. We also thank our colleagues Marin Litoiu from IBM Canada, Hausi Müller from the University of Victoria, Stefan Schuster from the European Software Institute, Soumya Simanta from the SEI, and Eleni Stroulia from University of Alberta for their valuable input.

Abstract

This report presents the results of the Foundations of Software-Oriented Architecture (FSOA) workshop held at the Third International Conference on Interoperability for Enterprise Software and Applications (I-ESA 2007). This workshop was organized to provide a forum for a concerted effort to develop a long-term, community-wide research agenda to bridge the gap between SOA research and the real needs of the practitioners in the field. An initial research agenda for SOA was presented along with three papers that focus on specific aspects of operations, engineering, and business challenges. The papers are each presented in this report, and the discussion and its implications for an evolving research agenda are summarized.

1 Introduction

Service-oriented architecture (SOA) has become an increasingly popular mechanism for achieving interoperability between systems. Standardization efforts are progressing, and a variety of tools are becoming available to support SOA development. Organizations, including banks, health care providers, and government agencies, are focusing on SOA as a way to reach a previously unachievable level of interoperability among their systems and agility within their business practices. At the same time, academic and industrial researchers are working on solutions to a range of relevant problems to address the needs of SOA adopters.

Significant progress is being made on several fronts; however, current research efforts seem to be proceeding in many directions in an uncoordinated fashion. There seem to be no clear, commonly agreed upon, overarching themes to focus research activity. As a result, there is a danger that important research needs will be overlooked while issues of peripheral long-term significance are investigated.

The Foundations of Software-Oriented Architecture (FSOA) workshop, held at Third International Conference on Interoperability for Enterprise Software and Applications (I-ESA 2007), was organized to begin developing a long-term, community-wide research agenda. This agenda is meant to bridge the gap between SOA research and the needs of the practitioners in the field. The workshop organizers brought together researchers and practitioners to identify a set of focused research themes to explore the foundations of SOA.

This report contains four SOA research papers. Section 2 of this report contains the paper that served as background for the workshop and a draft research agenda. Sections 3 through 5 contain additional papers that were presented at the workshop.[1]

- *A Proposed Taxonomy for SOA Research* by Kostas Kontogiannnis, Grace A. Lewis, and Dennis B. Smith. This paper presents the proposed taxonomy of SOA research issues organized into three areas or topics—business, engineering, and operations—plus an additional area of cross-cutting concerns (Section 2).
- *Towards Adaptive Service Engineering* by Daniel Schneider, Christian Bunse, and Klaus Schmid. This paper highlights the challenges and research issues in the context of engineering adaptable, service-oriented systems, and it represents an example of research on operations issues (Section 3).
- *Bridging the Gap Between Object-Oriented Programming and Service-Oriented Computing* by Sven De Labey, Marko van Dooren, and Eric Steegmans. This paper provides a basis for bridging the gap between the Java programming model and the SOA development paradigm and provides an example of engineering issues (Section 4).
- *Model Driven Development of Service Oriented Architectures—Transforming Business Logic into Service Infrastructures* by Xabier Larrucea, Gorka Benguria, and Stefan Schuster. This

[1] The papers in this report that were presented at the workshop appear as they did in the original presentations; they have not been edited further (aside from adjusting their section, footnote, figure, and table numbers for the layout in this report and modifying the styles for citations and references).

paper addresses the challenge that different stakeholders interpret information captured in models from their own specific viewpoint. It presents experiences in the application of a Model-Driven Service-Oriented Architecture (MDSOA) framework and discusses a set of research challenges. This paper is an example of business research issues (Section 5).

Section 6 presents a summary of the discussions of the workshop. These discussions have been folded into the evolving SOA research agenda.

2 A Proposed Taxonomy for SOA Research

Authors: Kostas Kontogiannnis,[2] Grace A. Lewis,[3] and Dennis B. Smith[4]

Abstract: Service orientation has been touted as one of the most important technologies for designing, implementing, and deploying large-scale, service-provision software systems. In this position paper, we investigate an initial classification of challenge areas related to service orientation and service-oriented systems. We start by organizing the research issues related to service orientation in three general domains—business, engineering, and operations—plus include a set of concerns that cut across those domains. We further propose the notion of service strategy as a binding model for the three domains. Finally, we outline a set of emerging opportunities to be used for further discussion.

2.1 INTRODUCTION

Over the past decade we have witnessed a significant growth of software applications that are delivered in the form of services utilizing a network infrastructure. These services are available either on corporate intranets or on the internet and are delivered on open or proprietary network protocols. This approach to systems development is commonly referred to as service-oriented architecture (SOA), SOA-based systems, or service-oriented systems.

In this context, there has been gradual evolution. The first generation of service-oriented systems was based on monolithic components that could be reconfigured at compile time. Currently, the second generation of service-oriented systems is based on vertically integrated components that can be adapted and reconfigured at installation time and to some extent at runtime. Trends point to an emerging third generation of service-oriented systems that is cross-vertically integrated, context-sensitive, and reconfigurable in an autonomic, ad hoc, and on-demand manner [Fitzgerald 2006].

The gradually increasing adoption of service-oriented systems is supported by the technical and business communities. From a technical perspective, service-oriented systems are an approach to software development where services provide reusable functionality with well-defined interfaces; a service infrastructure enables discovery, composition, and invocation of services; and applications are built using functionality from available services. From a business perspective, service-oriented systems are a way of exposing legacy functionality to remote clients to implement new business process models by utilizing existing or third-party software assets, reducing overall IT expenditures while increasing the potential for innovation through software investments [Bieberstein 2006]. From either perspective, and despite their initially slow adoption and the conflicting standards proposed to support them, service-oriented systems are becoming the *de facto* approach to bridging the gap between business models and software infrastructure and for flexibly supporting changing business needs [Marks 2006].

[2] National Technical University of Athens, kkontog@softlab.ntua.gr

[3] Carnegie Mellon® Software Engineering Institute (SEI), Integration of Software-Intensive Systems (ISIS) Initiative, glewis@sei.cmu.edu (Carnegie Mellon is registered in the U. S. Patent and Trademark Office by Carnegie Mellon University.)

[4] SEI, ISIS Initiative, dbs@sei.cmu.edu

It is clear that the SOA paradigm is having a substantial impact on the way software systems are developed. However, although significant progress is being made on several fronts, current efforts seem to evolve in many directions. As a research community that has gone through a substantial growth spurt, we find ourselves facing a great opportunity and challenge:

> *To better channel our research efforts, we should attempt to reflect upon our progress to-date, recognize how our efforts and results build on each other, and identify—and potentially prioritize—the areas that we still need to investigate.*

In this position paper, we provide an initial classification of research issues pertaining to the business, engineering, and operation aspects of service-oriented systems, plus a set of cross-cutting concerns.

- Subsection 2.2 presents the main theme areas for issues related to service orientation.
- Subsection 2.3 outlines the taxonomy of SOA research issues.
- Subsection 2.4 presents examples of research issues in each area.
- Subsection 2.5 presents some emerging opportunities for research in service orientation.
- Subsection 2.6 provides a summary and outlines next steps.

2.2 SOA DOMAINS

The term *service orientation* implies distinct sets of issues and activities to different audiences. For example, to software engineers, it is all about functional requirements, components, integration techniques, messaging, tools, development environments, and middleware. To business people, it is all about implementing business strategies, enabling leaner IT departments, facilitating agile process models, and driving new service-delivery processes. Finally, to software-system users it is all about transparency, flexibility, ubiquitous access to services, and most importantly applications that ease their lives (e-government, e-health, entertainment). In fact, the service-orientation domain is so diverse and multidisciplinary that the term Services Science has already been adopted by industry and academia to denote the systematic study of the variety of issues related to all these variants of service orientation [Chesbrough 2006, Horn 2005, IBM 2008]. Furthermore, investigating and assessing the supporting technology, best practices, and guidelines for designing, implementing, adopting, and using SOA systems have strongly emerged as important aspects in this area [Allen 2006].

In an ideal SOA adoption setting (represented in Figure 2-1), an organization develops a service strategy that takes into account its business drivers, context, and application domain (the problem space). The service strategy defines the rationale for adoption of SOA, as well as goals and objectives. It also identifies a set of plans to achieve these goals and objectives (the planning space). The execution of these plans requires decisions to be made in the domains of business, engineering, and operations, as well as a cross-cutting domain that has an effect on the other domains. The cross-cutting domain includes such areas as training, education, and governance. Together, the four research domains compose our solution space. This classification of domains builds upon the notion of a conceptual model of business as considered in the IBM *Business Process Definition Metamodel* [IBM 2004a].

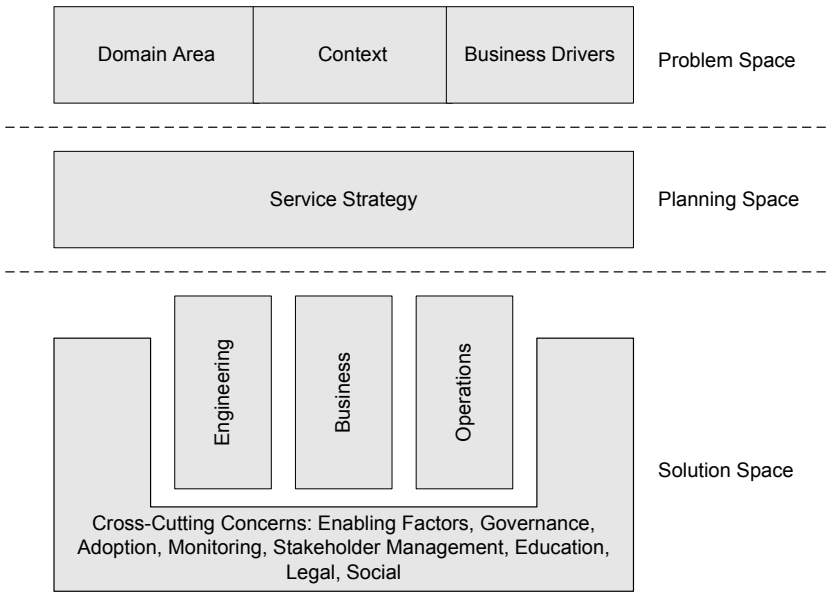

Figure 2-1: SOA Solution Space Domains and their Relationship to Service Strategy

2.2.1 Business Domain

The business domain deals with the form and the impact that service orientation may take in a given organization, application domain, or context. Some of the drivers for service-oriented systems in this domain are the needs to be on-demand, customizable, trusted, compliant, agile, and measurable. Aspects of this domain therefore include business processes vis-à-vis service orchestration and choreography, organizational and financial aspects, ROI evaluation of service orientation-related IT decisions, new business models, widespread information dissemination, business strategies, workforce management methods, and service performance management.

2.2.2 Engineering Domain

The engineering domain deals with the main aspects of the service-oriented system life cycle. Some of the drivers for service-oriented systems in this domain are the needs to be reliable, secure, open, robust, efficient, and testable. Aspects of this domain therefore include process models that can be used to build service-oriented systems, requirement models for denoting functional and non-functional aspects, platform- and computational-independent architectural abstractions, design patterns, logging, model-driven code generation, verification, testing, and maintenance.

2.2.3 Operations Domain

The operations domain deals with the operation, evaluation, and optimization of service-oriented systems. Some of the drivers in this domain are for service-oriented systems to be ambient, user-friendly, high impact, pervasive, and adoptable. This domain therefore includes any aspects related to the adoption, monitoring, and support of deployed service-oriented systems.

2.2.4 Cross-Cutting Concerns

There are issues that have an effect on all the domains. These issues include governance, training and education, social and legal aspects, and people skills.

2.2.5 Service Strategy

At the core of an SOA environment is a service strategy, which ties the business, engineering, and operations domains together, informing the decisions made in each of these domains and reflecting the influence they have on each other.

The properties and characteristics of a service strategy can be identified. Service strategy provides the

- cause-effect and impact links behind the decisions taken at the business, engineering, and operations levels in an organization. For example, the service strategy will identify the rationale for selecting a technical solution, given a business decision or an operational requirement.
- top-level flow of activities for the life cycle of a service-oriented system
- common ground for the analysis of a service-oriented system that takes into account different perspectives and points of view. For example, a service strategy will provide the rationale for evaluating a system from a technical perspective given the specific constraints of its business and operational context.

Figure 2-2 depicts a service strategy at a high level. A service strategy is developed taking into consideration the business drivers for SOA adoption, the domain in which the organization operates, and the context for SOA adoption (such as budget, schedule, and policies).

A plan for the execution of the strategy is formulated, taking into account the business and service models for the organization. At a very high level, the business model is the representation of an organization's business processes, and the service model is the representation of an organization's existing services and how they relate to the business processes in the business model. The plan is executed and a service-oriented system is deployed. Data is collected for the evaluation and optimization of the service-oriented system and fed back into the formulation of the strategy and plans, thus reflecting the evolutionary nature of service-oriented systems.

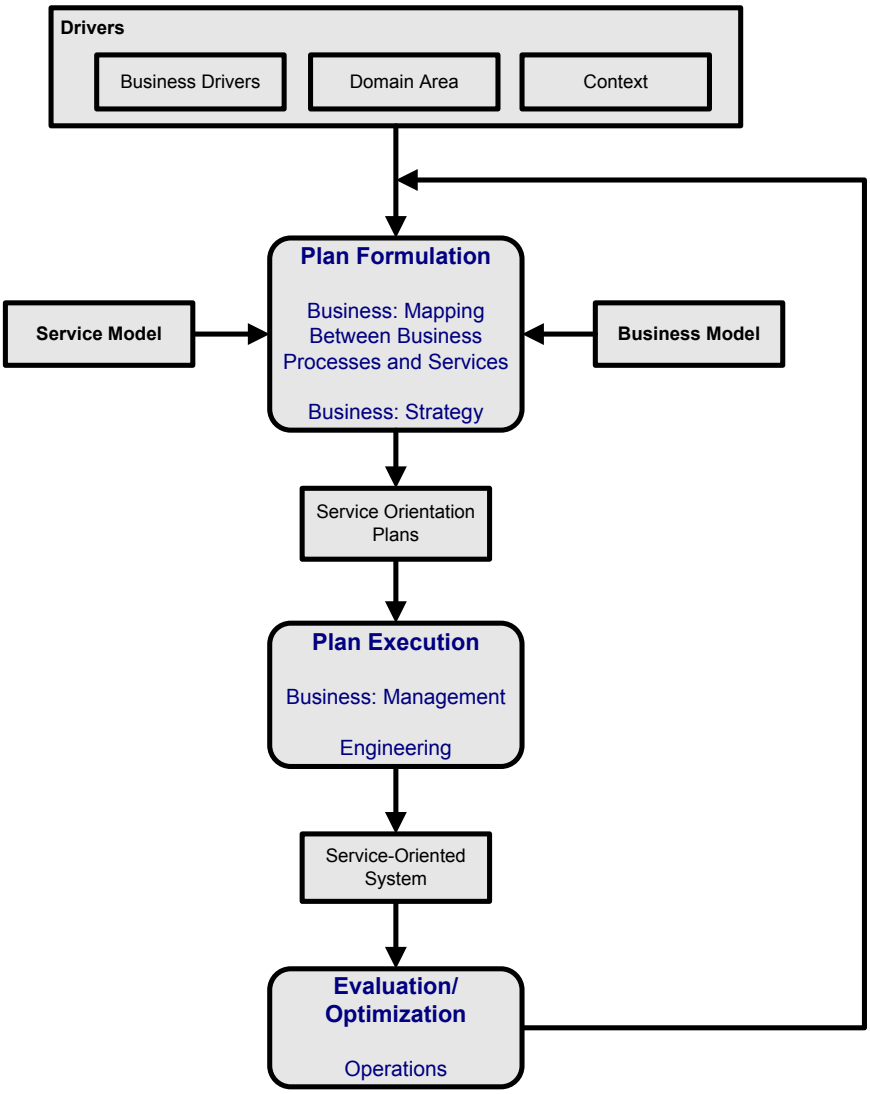

Figure 2-2: Service Strategy

2.3 RESEARCH CHALLENGES FOR SERVICE-ORIENTED SYSTEMS

In this section, we offer an initial classification of the research issues related to service orientation. The challenges listed under each category are still at a very high level. They are based on a preliminary literature search, expert opinions from academia and industry, as well as the authors' experience. This list is by no means complete, and it is the intent of the authors to gather feedback from a wide community through exposure of the proposed classification and challenges.

2.3.1 Business Domain Research Topics

Strategy: How are SOA strategies matched to business goals? How are they justified?

- techniques to establish and document the business case for service orientation
- techniques for strategy selection

Mapping between Business Processes and Services: How are services mapped to business processes?

- techniques for service identification
- techniques and processes to support the strategic reuse of legacy components and services
- analytic methods for service evaluation—that is, evaluate how well a given service fits business needs
- techniques and processes for establishing relations between business and service models
- investigation of industry and domain-specific standards for business and service modeling
- processes to support the adaptation of services to meet changes in business processes

Management: How to manage service-oriented projects?

- models for workforce allocation—alignment of people across organizations to implement agile business processes; collection and analysis of data for workforce management and productivity optimization
- investigation of models for contract pricing and negotiation in an on-demand service setting
- models for organizational structures in SOA environments
- patterns and stereotypes for roles and responsibilities of the involved stakeholders
- investigation of the relation between IT metrics and business metrics
- methods, models, and representations for assessing the effectiveness of services
- use of operations research concepts and techniques in service-oriented environments

2.3.2 Engineering Domain Research Topics

Process and Life Cycle: How do processes and system life cycle change in a continuous running system in a distributed setting, with distributed and diverse stakeholders?

- development processes and methodologies for service-oriented systems—e.g., adaptation of existing methods, feature-oriented software development, and role of Component-Based Software Engineering
- service-oriented system life-cycle issues—such as managing a continuous running system in a distributed setting with distributed and diverse stakeholders

Requirements: How does one model manage evolvable and potentially conflicting requirements between different perspectives?

- modeling and management of nonfunctional/soft/evolvable and possibly conflicting requirements
- requirements specification, modeling, and management from different perspectives: service provider, service consumer, and infrastructure provider

Service Selection: Once candidate services are identified, how does one select the *best* service?

- service selection criteria—e.g., reliability, performance, and risk
- service evaluation techniques
- models to support strategic reuse

Service Definition and Categorization: What are best practices for service definition and categorization?

- taxonomy and repositories of best practices for the definition and classification of services
- guidelines for design decisions—e.g., appropriate granularity of services to be used
- techniques and models for the syntactic and semantic description of services

Architecture and Design: What are best practices for the architecture and design of service-oriented systems, especially in distributed, third-generation environments?

- modeling the dynamic runtime architecture of a service-oriented system
- features and properties for next-generation frameworks for development of service-oriented systems
- architectural styles for service-oriented systems
- architectures for data integration in service-oriented environments—e.g., mediation, consolidation, ownership, semantics, and metadata
- communication and connectors—synchronous, asynchronous
- styles and design decisions that favor certain nonfunctional requirements—e.g., design for security and design for performance
- architectures for service types—including data services (information as a service), business services, and infrastructure services
- design patterns for service-oriented systems and best practices of existing and new design patterns
- design for personalization, context awareness, and adaptation—e.g., time zones, language
- design for runtime, semantic-based discovery, and composition
- relationship between product lines and service-oriented systems
- protocol mediation and wrapping in multi-protocol environments

Implementation: What are best practices for implementing service-oriented systems?

- model-driven approaches and template-based code generation
- language extensions to support service-oriented development
- transactional support, exception handling, and compensations

Tools and Products: What are available tools and tool characteristics to support the engineering of service-oriented systems?

- integrated development environments to support service-oriented development—features such as model refactoring and incremental model synchronization
- evaluation guidelines for tool and product support
- support for service choreography and orchestration
- collaboration tools in distributed development environments
- service appliances—e.g., XML processing and security

Testing: How does one test at multiple levels in dynamic environments, including testing of elements that are potentially outside of an organization's control?

- infrastructure testing—e.g., messages, specifications, model properties, and deployment descriptors
- functional service testing—white-box and black-box testing of services that considers all potential compositions
- integration testing—including transaction management, quality of service, load/stress testing, composition, workflow simulators, orchestration, versioning, monitoring, and regression testing
- system testing in dynamic environments
- simulation and what-if analysis testing
- acceptance testing—possibly more diverse than typical acceptance testing
- practices for service providers to support system testing of their consumers
- test beds and benchmarking

Deployment: What are techniques to ease the deployment of services in heterogeneous platforms, especially in third-generation systems?

- pre-start checks to ensure that all required components and configurations are properly set
- techniques for multi-platform support configurations
- techniques for context and location awareness support

Maintenance and Reengineering: What does maintenance and reengineering look like in a dynamic, heterogeneous, and potentially distributed development and maintenance environment?

- evolution patterns
- tools, techniques, and environments to support maintenance activities, including dependency and impact analysis, change control, and management
- multilanguage system analysis and maintenance
- reengineering processes
- tools for the verification and validation of compliance with constraints
- round-trip engineering in service-oriented systems

2.3.3 Operations Domain Research Topics

Adoption: What can ease the adoption of service-orientation for a service consumer?

- investigation of requirements and types of portals and related collaboration environments
- analysis and models of customer adoption processes
- investigation of methods for evaluating and assessing service usability
- issues and models related to the adoption and advertisement of services, from the perspectives of service and infrastructure providers

Monitoring: To provide feedback for SOA strategy, what does one measure and how does one measure it in SOA environments? How can monitoring be used for runtime adaptation?

- techniques for monitoring of business processes in an SOA environment, including instrumentation and logging
- operations monitoring for auditing purposes
- models of self-healing systems—runtime diagnostics and dynamic-reconfiguration methods
- techniques for resource allocation and configuration management in an SOA environment—e.g., virtualization and runtime provisioning

Support: How does one provide service consumer support in deployed service-oriented systems?

- techniques and methods for operations support in an SOA-enabled organization—e.g., problem management
- techniques and models for the specification and dissemination to all stakeholders of the appropriate service-level agreements

2.3.4 Cross-Cutting Research Topics

Governance: How does one model and implement SOA governance?

- techniques and processes to model policy, risk, and trust and to ensure that a service acts on requests that comply with claims required by policies
- investigation and compilation of repositories and guidelines of best practices for compliance monitoring in given domains

Social and Legal Issues: What are the social and legal effects of service orientation?

- social and legal implications related to the deployment and use of services
- legal compliance
- roles of service orientation in social computing

People Skills/Capital: What skills and resources are needed for successful implementation of service-oriented systems?

- skills required to develop, use, and maintain a service-oriented system
- matching needed services with user skills and abilities

Application Domains: Are there service-orientation aspects that are specific to certain domains?

- investigating application domains and potential impact, such as health care, government, finance, banking, electronics, telecommunications, and automotive
- establishing, where appropriate, industry and domain-specific standards for data, services, business processes, best practices, and performance indicators

Enabling Factors: How does one enable SOA adoption?

- analysis of technology issues as an enabler for service orientation

Stakeholder Management: How does one manage so many diverse stakeholders?

- techniques to mediate different and diverse stakeholders' needs, requirements, and views

Training and Education: What do we need to teach people to deal with service orientation?

- establishing the area of services science
- investigating and developing appropriate university curricula

2.4 A RESEARCH AGENDA FOR SOA—SELECTED TOPICS

A goal for this work is the publication of an SOA Research Agenda that will become a source of topics for researchers in industry and academia to advance the state of the practice and the rate of adoption of service orientation as a systems development paradigm.

As an example of the work in progress, we have selected several topics for further elaboration in the following subsections. For each research topic, we give a rationale to indicate why it is important. Then, we present a set of references under current efforts as examples of work being done in each area. Finally, we present specific the challenges and gaps in each research area.

2.4.1 Examples of Business Research Topics

2.4.1.1 Strategy Selection

Rationale

SOA provides the potential of enabling greater cost-efficiency, agility, adaptability, leveraging of legacy investments, and interoperability between systems and organizations. However, the wrong strategy can result in an expensive collection of random services that are never used. Table 2-1 shows some examples of how different business needs and goals lead to different SOA strategies.

Table 2-1: Examples of Business Needs and SOA Strategies

Business Needs and Goals	Focus of SOA Strategy
Increase information available to customers	• Easy-to-use and customizable portal • Services that provide information of interest for customers
Integrate business partners	• Infrastructure to support integration with business partners potentially running on different computing platforms • Back office integration • Identification of business rules
Improve internal business processes	• Identification of key business processes • Elimination of redundancy • Services that access legacy systems

Current Efforts

Regardless of the business goals it supports, a successful SOA strategy should include [Lewis 2007]

- evidence of fulfillment of critical business goals
- alignment with organizational enterprise architecture and current and future information technology (IT) infrastructure
- realistic choices of technologies and infrastructures
- realistic and gradual adoption strategy
- adequate SOA governance structure

- priorities for implementation
- reuse strategy across internal and external organizations

Case studies and vendor methods (e.g., IBM, AgilePath, and Software AG) are converging on the need for establishing a direct connection between business strategy and SOA strategy. However, in practice, the lines between SOA strategy and SOA governance are blurred, resulting in a focus on the "how to implement" rather than the "what to implement," which can result in pressure for developing solutions that may be sub-optimal. As an example, there is more work on how to model business processes rather than on how to determine which business processes provide the greatest potential for modeling.

Challenges and Gaps

Important research topics in this area are those that address the identification of appropriate SOA strategies, such as

- a set of SOA strategy patterns that map abstract business goals and needs to potential SOA strategies or elements of a SOA strategy
- guidelines for iterative selection of services based on business goals and needs as well as metrics collected from deployed services

2.4.1.2 Business Case for SOA

Rationale

In general, there is recognition that SOA adoption can provide business agility, adaptability, legacy leverage, and integration with business partners. Given these goals, an important criterion for making business decisions concerns the amount of investment that is required for SOA adoption and the projected payoff over a certain period of time.

Current Efforts

There is current work that identifies the business value of SOA adoption in various industries, as indicated by references such as

- Tilley et al discuss the business value of web services when used for enterprise application integration or business-to-business (B2B) commerce [Tilley 2004].
- Brandner et al claim enhanced integration capabilities of its core banking system through the use of web services [Brandner 2004]. There are other Australian, U. S., and Finnish success stories in the banking industry.
- Pujari discusses the pros and cons of self-service technology, potentially enabled by SOA technologies, in the Canadian B2B industry [Pujari 2004].
- Linthicum has written several articles on the subject, including one in which he proposes a formula for calculating the relative value of SOA adoption [Linthicum 2006].

There are other case studies and articles that provide anecdotal evidence of the business value of SOA adoption. Many of these studies are sponsored by vendors or co-authored with vendors. The problem is that these are examples of point solutions that make it difficult to generalize across organizations. A comprehensive framework for understanding the business value of SOA has not yet been developed.

Challenges and Gaps

The largest gap in this area is that more vendor-neutral data needs to be gathered. Current efforts have focused on individual case studies and there have not been rigorous analyses that can be generalized. This triggers another important question: How does an organization measure common benefits associated with SOA adoption, such as business agility, legacy leverage, or increased interoperability? An important research focus is to gather data from both success stories and failures, find commonalities, and start to develop a framework for calculating the business value of SOA adoption.

2.4.1.3 Organizational Structures

Rationale

Moving toward a service-oriented environment is a cultural shift for organizations, especially for those that have not had a business process focus. It will most probably require restructuring of business as well as IT organizational structures, and this restructuring will drive roles and responsibilities, budget, and development.

Current Efforts

There is recognition that service-oriented systems require different types of organizational structures and roles that enable alignment between business and IT.

- Bieberstein et al propose an organizational model that enables cross-business unit teams [Bieberstein 2005].
- Balzer suggests the creation of an architecture office responsible for the responsible for the alignment between SOA efforts and the business requirements [Balzer 2004].
- McClure proposes a matrix structure between service managers and owners that will the goal of providing service delivery and support that is aligned with business goals and objectives [McClure 2006].

Research efforts have begun to specifically identify roles and responsibilities in service-oriented systems development [Kajko-Mattsson 2007, Kajko-Mattsson 2008, Pai 2007]. However, there is very little documented evidence in the public domain that organizations have gone far along the "SOA path." Most case studies are pilot efforts or point solutions that describe a single experience rather than an adoption of SOA as a complete approach to systems development that would require changes to organizational structures.

Challenges and Gaps

There is a need for more concrete guidance on proper organizational structure for service-oriented development that goes beyond the recognition of the need for business and IT alignment and cross-functional teams, especially in cases when services will be consumed across various lines of business (LOB).

- How does having shared services change the existing organizational structure inside an enterprise?
- Who funds these shared services?
- Who owns these shared services and is responsible for creating and maintaining them?
- Are there variations between effective organizational structure by organization type, domain, and culture?

Best practices and guidelines that are based on success stories (in specific domains) are needed so that they can be leveraged by other organizations.

2.4.2 Examples of Engineering Research Topics

2.4.2.1 Development Processes and Methodologies

Rationale

Development processes and methodologies for service-oriented systems have to deal with a very dynamic SOA environment—the market is dynamic, business processes are dynamic, and technology is dynamic. What this means is that service-oriented systems are in constant evolution, to the point that the term "perpetual beta" is used to describe the state of a service-oriented system. Development processes and methodologies have to be capable of dealing with this dynamicity.

Another aspect that characterizes development in this environment is that there should be a "discover and reuse" mentality instead of a "build from scratch" mentality. Development processes and methodologies should enable reuse.

Current Efforts

IBM has defined a life cycle for SOA-based systems that is practice-based and consistent with other work on SOA life cycle [Borck 2006, High 2005, Rodriguez 2005, Veryard 2004]. This SOA life cycle, presented in Figure 2-3, consists of the following phases: modeling, assembly, deployment, and management.

- *Modeling* is the process of capturing business requirements, business goals, and objectives and transforming them into business process specifications—the business model. The modeling phase also includes the analysis of the model through what-if scenarios applied to the business processes.
- *Assembly* deals with implementation issues: the business models are implemented by reusing existing services or creating new services. Functional testing is part of this phase.
- *Deployment* includes service dependency resolution, capacity planning, hosting infrastructure definition, and system testing.
- *Management* refers to the operational activities that keep the applications running, as well as to the measurement of IT and business performance indicators, logs, and traces for auditing and feedback for other phases of the SOA life cycle.

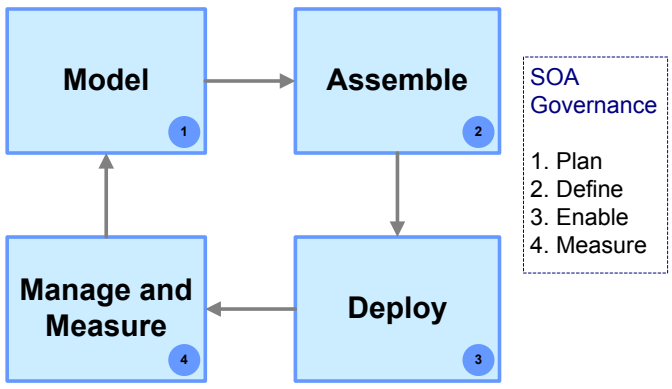

Figure 2-3: SOA Life Cycle Mapped to SOA Governance Phases

The instantiation of the SOA life cycle should be done in the context of what is known as SOA governance, as also shown in Figure 2-3 [Brown 2006, Windley 2006]. SOA governance is the process of establishing the chain of responsibilities and communications, policies, measurements, and control mechanisms that allow people to carry out their responsibilities. SOA governance itself has a set of phases: plan, define, enable, and measure. (Challenges associated to SOA governance can be found in Section 2.4.4.1.)

1. The *Plan* phase documents the existing IT capabilities and defines a governance plan.
2. The *Define* phase defines or modifies the governance processes and the governance infrastructure.
3. The *Enable* phase deploys the governance mechanisms and infrastructure, as well as the policies.
4. The *Measure* phase monitors the compliance with policies and the effectiveness of the governance.

SOA governance typically includes policies and procedures, roles and responsibilities, and design-time and runtime forms [Gold-Bernstein 2006].

Given the focus on business requirements and processes as well as the need to incorporate governance, there is recognition of the differences between traditional development and service-oriented systems development, as indicated by several terms currently used to describe development in SOA environment:

- Policy-Driven Development approaches focus on following policies through to deployment.
- Business-Driven Development uses business requirements to drive service design and construction.
- Business-Process-Oriented Development uses business processes to drive service identification, development, and deployment.

Application life-cycle management and integrated development environments are starting to incorporate SOA life-cycle elements.

Challenges and Gaps

The major gap in this area is that there is characterization of software development processes for SOA environments, but very few concrete processes (i.e., most of the focus is on what needs to be done but not on how to do it). Ziemann et al express this problem as follows:

> *The reason for this reluctance [refers to SOA adoption reluctance] is the existence of serious gaps in current SOA development methods: they describe how to technically implement a SOA on a green field, and do not describe how business requirements influence this development, nor do they pay enough attention for integrating the characteristics of existing systems* [Ziemann 2006].

Another problem is that some work considers the case of multi-organization development, where there is separation between service provider and consumer, but most development processes are targeted to single-organization development environments. There have to be processes that accept this emerging trend and include elements that allow for asynchronous development of service consumers and service providers belonging to different organizations. An example of one of these elements is service mocks [Woolf 2005]. Using service mocks is a simple technique that takes use

cases and converts them to tests, which then are used to create service mocks (mock objects) that pass the tests. Mocks are shared between service provider and consumer.

2.4.2.2 Design for Runtime Service Discovery and Composition

Rationale

Runtime discovery and composition of services can only be possible with semantically described services. A significant area of current work and research in dynamic binding is that of Semantic Web Services (SWS), which uses a markup language that is descriptive enough for a computer to obtain the information it needs to discover, compose, and invoke web services without human intervention. SWS are usually described using concepts from an ontology to provide the shared semantics between service provider and service consumer.

Current Efforts

There are multiple ongoing efforts for the semantic description of services. Four different recommendations have been submitted to the Worldwide Web Consortium (W3C):

- OWL Web Ontology Language for Services (OWL-S)
- Semantic Web Services Framework (SWSF)
- Web Service Semantics (WSDL-S)
- Web Service Modeling Ontology (WSMO) [W3C 2004a, W3C 2005a, W3C 2005b, WSMO 2008].

The Semantic Web Services Interest Group was created to charter the integration of this work. This group produced a recommendation for Semantic Annotations for WSDL and XML Schema specification [W3C 2007].

Challenges and Gaps

This is a very difficult, unsolved problem. The greatest challenge is that there is still not a standard for SWS. The tool support for current efforts is weak, and there are few examples of industrial use. One of the main challenges, specifically related to service design, is the granularity at which services are to be described and thus the precision that can be achieved during discovery [Küster 2007].

2.4.2.3 Support for Context Awareness

Rationale

In a context-aware SOA environment, services can be selected and adapted every time, according to the user and invocation context requirements and profiles:

- provision of a service with different performance, reliability, or security characteristics according to who invokes the service and from where it is invoked
- provision of a service that returns information based on user's language, time zone, invocation environment

Current Efforts

There are several service discovery mechanisms in use today:

- DHCP (Dynamic Host Configuration Protocol)
- UPnP (Universal Plug and Play)
- SLP (Service Location Protocol)

- Jini, UDDI (Universal Description, Discovery and Integration)
- X.500 [Droms 2002, UPnP 2008, Kempf 1999, Jini 2007, UDDI 2008, X500 2008]

Unfortunately, none of these mechanisms enables location-based services, semantic discovery, adaptive services, or dynamic composition. WS-Context is an OASIS 2007 specification with the goal of "providing a definition, a structuring mechanism, and service definitions for organizing and sharing context across multiple execution endpoints" [OASIS 2007]. Even though the specification refers to execution context and not consumer context, some of the extension points could be used to provide this type of information to a web service.

Advances in semantic service descriptions have the potential for supporting context awareness because they could be used to increase precision. In Section 2.4.2.2, we talk about some of the work, challenges, and gaps in this area.

Challenges and Gaps

There is a need for significant research on how SOA infrastructures can enable service discovery that is

- adaptive: Service changes its characteristics according to the context in which it is used or invoked.
- semantically based: Service is described in terms of its behavior not its name or signature or API.
- context-aware: Different services may be selected in a given time according to the context (who/where/time) the service is accessed. The following questions have to be answered in order to support context awareness in SOA environments:
  - What is context?
  - How can context best be modeled and represented?
  - How can semantic description of services be used to facilitate context-aware discovery?
  - What are the SOA infrastructure requirements in a context-aware service provision environment?

2.4.2.4 System Testing

Rationale

In an SOA environment, system testing means end-to-end testing. The problem is that in SOA environments, systems components are distributed, deployed on heterogeneous platforms, and often not even available.

Current Efforts

The market for tools to test for SOA environments (mainly for web services) is growing. Tools are available to perform testing at multiple levels—from business processes to messages—as well for qualities such as availability, performance, and security. However, most testing tools are incapable of building composite interdependent tests across technology platforms, languages, and systems. Also, most testing tools assume control over all elements of the service-oriented system. Sometimes client developers typically only have access to interfaces (e.g., Web Services Definition Language [WSDL] description files in the case of Web Services) and lack access to code. This has triggered some research into the use of gray-box testing, which is appropriate when there is limited knowledge.

Challenges and Gaps

The challenges in system testing are driven by the distributed, heterogeneous nature of service-oriented systems components and a growing market of third-party services, which means that there is not a single owner of the complete system. This triggers some challenging research topics such as

- dynamic testing in distributed, heterogeneous environments
- service certification
  - What does a certification process look like?
  - What can be certified?
- enhanced service repositories that provide test cases for services
  - How are test cases specified?
- test-aware interfaces for service consumers to test services
  - Given that providers would need to have test instances of services, how are these testing services specified, and how do service consumers become aware of their existence?

We also need to recognize that it is not always possible to do end-to-end testing. In such cases, interesting research topics are

- simulation of service-oriented system environments
- best practices for exception handling

2.4.2.5 Reengineering Processes

Rationale

Because it has characteristics of loose coupling, published interfaces, and a standard communication model, SOA enables existing legacy systems to expose their functionality as services, presumably without requiring that significant changes be made to the legacy systems. Migration of legacy assets to services has been achieved within a number of domains—including banking, electronic payment, and development tools—showing that the promise is beginning to be fulfilled. While migration can have significant value, any specific migration requires a concrete analysis of the feasibility, risk, and cost involved. The strategic identification and extraction of services from legacy code is crucial as well.

Current Efforts

There are not many reengineering techniques that focus on a "full-circle" model; one is the SOA-Migration Horseshoe proposed by Winter and Ziemann and shown in Figure 2-4 [Winter 2007]. This approach integrates software reengineering techniques with business process modeling and recommends applying reverse engineering techniques to extract a Legacy Enterprise Model from the legacy code. Then, applying enterprise modeling techniques, a Consolidated Enterprise Model is created, from which services are identified using forward engineering techniques. Finally, legacy code is mapped to services via wrapping or transformation.

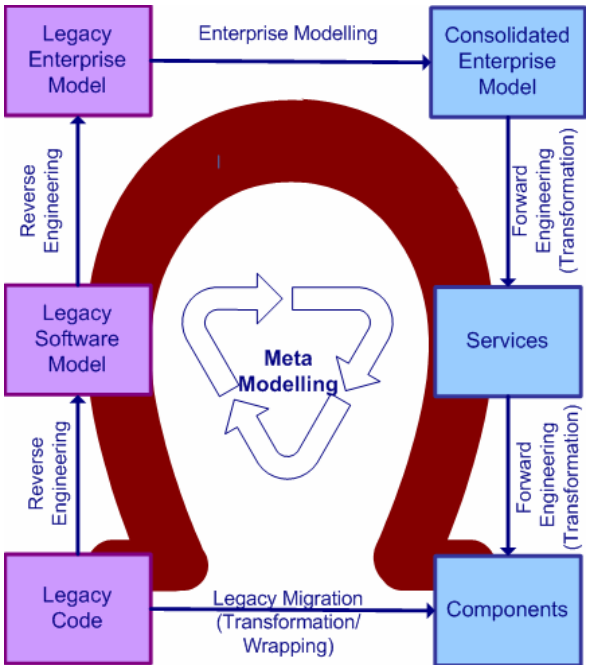

Figure 2-4: SOA-Migration Horseshoe [Winter 2007]

The larger amount of work is on techniques in the bottom portion (the open end) of the horseshoe for exposing legacy functionality as services, mainly Web Services [Chawla 2007]. Tools to support this type of migration are available as language libraries or integrated into common Integrated Development Environments (IDEs) such as the Eclipse Web Tool Platform (WTP) and the .NET development environment, or as part of infrastructure products such as Apache Axis [Eclipse 2008, Shodjai 2008, Apache 2005].

Some work on techniques and research proposals that takes into consideration business goals and drivers when making migration decisions to an SOA environment; that is, it is consistent with a strategic approach to SOA adoption. These techniques work in the "top portion" of the horseshoe.

- The Service Migration and Reuse Technique (SMART) is a method for determining the feasibility of migrating legacy systems to a SOA environment, which takes into consideration business drivers as well as characteristics of the legacy system [Lewis 2007]. The output of this method is a migration strategy that includes preliminary estimates of cost and risk and a list of migration issues.
- Ziemann et al agree that "rather than being of a purely technical nature, the challenges in this area are related to business engineering: How can a sub-functionality be identified as a potential service, or how can business process models be derived from a legacy system." They propose a business-driven legacy-to-SOA approach based on enterprise modeling that considers both the business and legacy system aspects [Ziemann 2006].
- IBM offers a method called Service Oriented Modeling and Analysis (SOMA) that focuses on full system development but has some portions that address legacy reuse: "SOMA facilitates integration with techniques for analyzing legacy applications, custom and packaged, to identify, specify and realize services for use in a service-oriented architecture. It breaks out the business functions of each existing application, identifying candidate services that can be used to realize business goals under the new architecture. It also identifies potentially proble-

matic areas and highlights areas where new services need to be developed or sourced from an external provider." Unfortunately, there is not much public information on the method [IBM 2005]. Not surprisingly, most major vendors have embraced the migration of legacy systems to SOA environments and offer services in this area.

- Cetin et al propose a mashup-based approach for migration of legacy software to pervasive service-oriented computing platforms [Cetin 2007]. The interesting aspect about this work is the inclusion of presentation services, which is not typical. The approach combines two views: top-down, starting from business requirements, and bottom-up, looking at legacy code. Business requirements are mapped to services and integrated through a mashup server, which eliminates the need for developing specific applications to access the services.

Finally, there is work related to the identification of services in legacy code, addressing the "left portion" of the horseshoe.

- In the context of web services, Aversano et al propose an approach that combines information retrieval tracing with structural matching of the target WSDL with existing methods.[5] It performs first library schema extraction and then feature extraction to build a WSDL document from the legacy code. Then, it compares the generated WSDL document with the target WSDL document using structural matching.
- Also in the context of web services, Sneed proposes an approach that consists of salvaging the legacy code, wrapping the salvaged code, and making the code available as a web service [Sneed 2006]. In the salvaging step, he proposes a technique for extracting services based on identifying business rules that produce a desired result.

Challenges and Gaps

The ideal reengineering process would be one that implements the SOA-Migration Horseshoe. The problem is that there are currently techniques and tools that implement portions of the horseshoe but not the full horseshoe. An important area of research would be the development of concrete processes that implement the horseshoe and tools (or suites of tools) to support the process. The automation of this process would be a very complex task that is worth investigating.

Also, as stated by most of the people doing work in this area, the real challenge is mining legacy code for services that have business value. Research topics in this area include

- tools and techniques for analyzing large source code bases to discover code that is of business value
- metrics for "wrapability" and business value to determine reusability
- application of feature extraction techniques to service identification, given that services usually correspond to features [Sneed 2007]

[5] This proposal was made in a paper called "Identifying Services from Legacy Code: An Integrated Approach." Aversano, Di Penta, and Palumbo presented the paper at the working session on Maintenance and Evolution of SOA-Based Systems (MESOA 2007), held on October 4, 2007 in Paris, France, in conjunction with the 23rd IEEE International Conference on Software Maintenance (ICSM 2007).

2.4.3 Examples of Operations Research Topics

2.4.3.1 Service Level Agreements

Rationale

A service-level agreement (SLA) is a formal and bilateral contract between service provider and consumer to specify the requirements and uses of specific services. An SLA is essential for establishing trust between service providers and service consumers. In a growing third-party service market, SLAs can be used to differentiate and select from various available services, help service providers anticipate demand and plan their resource allocation accordingly, and serve as a mechanism for risk mitigation.

Current Efforts

There is a need for standardization, specification, and guidance for using SLAs in an SOA context. Web Service Level Agreements (WSLA) is a specification and reference implementation by IBM that provides detailed SLA specification requirements for enabling the monitoring of SLA compliance, guidance on how these requirements are addressed in the WSLA specification, and a WSLA framework that allows monitoring of SLAs at runtime [IBM 2007a]. CBDi provides basic guidance on how to approach SLAs from service consumer and service provider perspectives at a higher level than WSLA [CBDi 2006]. Given that most SLAs are based on specifying the required quality of service (QoS) for a service, an active research area is modeling and implementing various QoS attributes in service-oriented and dynamic environments.

Challenges and Gaps

An important contribution to SOA adopters would be the creation of a generic and standardized framework for SLA management across enterprises as well as lines of business inside an organization. This would involve providing appropriate automation and support for (1) mapping contractual SLAs to standard and actionable implementations and (2) monitoring and managing service level at runtime. In the area of QoS, more work needs to be done in understanding QoS of composite services, especially when lower level services in a composite service are provided by different providers.

2.4.3.2 Service Usability

Rationale

A market of third-party service brokers and providers is emerging, as shown by companies such as Amazon Web Services (www.amazon.com), SEEC Inc. (www.seec.com), the IBM SOA Business Catalog (http://catalog.lotus.com) and StrikeIron (http://www.strikeiron.com). In this service market, the characteristics that can make a service more or less attractive can include capabilities, SLAs, and usability. The characteristics that make a service more usable or less usable can include interface design; options in messaging protocols, language, and others; add-ons such as test cases and test instances; and any other metadata that can tell consumers more about the service [Houlding 2007].

Current Efforts

The concept of service usability and user-centered service design is starting to emerge [Abascal 2006, Hai 2006, Martens 2003]. Regarding service usability, many guidelines for service interface design have been published. However, good service interface design does not necessarily map directly to usability (i.e., a very good but complex service may not promote usability). User-centered service design is the involvement of the service consumer in service interface design,

similar to work of Nielsen, Constantine, and others on user-centered design and usability techniques [Nielsen 1993, Constantine 2000]. However, the ability to establish user-centered services is more complex in large organizations or groups of organizations where service developers do not have access to service consumers.

Challenges and Gaps

Challenges remain in this area, such as

- What characterizes usability in a service-oriented context?
- What lessons learned from user interface design and user-centered design can be applied to service interface design?
- What does the service market look like? What are organizations in that market looking for?
- How can these issues be captured and embedded in best practice for service engineering?

2.4.4 Examples of Cross-Cutting Research Topics

2.4.4.1 SOA Governance

Rationale

An InfoWorld 2007 SOA Trend Survey indicates that lack of governance is the main inhibitor for SOA adoption [InfoWorld 2007]. Effective SOA governance requires rules that define roles, responsibilities, and appropriate use of standards; make the expectations of a diverse set of stakeholders explicit; provide for SLAs; and monitor compliance through metrics and automatic recording and reporting.

Current Efforts

A number of organizations such as IBM, AgilePath, and Software AG have developed sophisticated models of SOA governance. These models focus mostly on relationships to corporate enterprise architecture, use of registries, SOA life-cycle management, SLAs, metrics on policy enforcement, effectiveness of services, and use of services. A number of tools have also begun to automatically incorporate metrics and aspects of governance.

Challenges and Gaps

Most efforts to define and implement governance are driven by vendors and guided by the governance aspects that can be automated by their tools. As with the business case for SOA, most case studies are anecdotal and idiosyncratic. An interesting research topic would be to establish an abstract model for SOA governance and its variations within different domains. A starting point could be the establishment of SOA governance elements, similar to what has been done at Hartford Inc. and the creation of a template, similar to some work done by Burton Group [Afshar 2007, Manes 2007].

2.4.4.2 Education: Services Science

Rationale

Services science is a concept that merges technology with an understanding of business processes and organization, an identification of critical problems, and the tools that can be applied to correct them [Chesbrough 2006]. The term was coined by IBM and applies to services in general. It is becoming accepted in industry and the academic world as an area of active research called SSME (Services Science, Management and Engineering) [IBM 2008].

Current Efforts

SSME educational programs have begun at universities in the United States, Canada, Brazil, the United Kingdom, Finland, Sweden, Israel, China, and Hong Kong. The scope of SSME goes beyond that of service in the SOA context. The term service is so broad that SSME programs are finding it difficult to establish a focus; therefore, they tend to generic. To go beyond the basics, Chesbrough and Spohrer pose the need for a unified model for service innovation as a challenge for services science, and Spohrer and Maglio call for a research agenda to understand service system evolution [Chesbrough 2006, Spohrer 2006].

Challenges and Gaps

The major challenge for training and education related to SOA is how to fit services science topics into already full curricula in computer science, business, and IT programs. Applicable skills include business and information technology, as well as the human factors that go into a successful services operation. Questions that need answers include

- What SOA-specific aspects can be extracted from services science, and how can they be operationalized?
- Into which programs are services science topics most appropriately placed?
- Are new, cross-disciplinary programs needed?

2.5 OUTLOOK—EMERGING OPPORTUNITIES

There are conflicting views with respect to the usefulness of service-orientation and its related implementations and standards (e.g., web services). At one extreme, service-orientation proponents advocate it as the solution to addressing the needs of the next generation software applications. At the other extreme, service-orientation non-believers advocate that it is just hype that has not delivered on promises and has been plagued by limited adoption by the industry. We believe that the truth is somewhere in the middle. On one hand, service orientation is a very promising paradigm for large systems. On the other hand, it is useful and should be considered if and only if there is a proven need, business case, and demand for it. We advocate that service orientation is and should be business-driven and demand-driven.

Service-orientation is not the solution to all problems. There are certain types of problems—such as interoperability, integration, choreography, context awareness, and design of ultra-large-scale distributed, service-based applications—where service-orientation can be of benefit. There are other types of problems and contexts where the technologies that support service orientation are not enough, or where the investment is too large to justify its implementation. We are also experiencing a healthy growth of technologies that are taking us gradually to the third generation of service-oriented systems, where context sensitivity, adaptivity, and "autonomicity" become important requirements.

Overall, the outlook for service orientation is very positive and with it come the challenges to support this paradigm, as well as the opportunities for new research. The quest for third-generation service-oriented systems will also provide an interesting set of challenges.

2.6 CONCLUSIONS

In this paper, we provide an initial classification of research issues in three domains—business, engineering, and operations, plus a set of cross-cutting research topics. Even though our thoughts

have evolved since the execution of this workshop and the taxonomy has changed based on feedback from this and other workshops, the essence of the research topics remains the same.

Despite the slower than desired adoption of service-oriented systems in industry, we believe that the outlook is very positive, provided that service orientation is always considered and applied within specific contexts and in order to solve problems that are suited for the technologies supporting it. Furthermore, we argue that for service orientation to succeed there must be a demand and a strong business case for it. There is a large amount of initial research in the engineering and operations domains and less in the business domain. This finding in itself exposes a large potential for research, if we believe that service-orientation adoption should be business- and demand-driven. Hence, we note the need for a service strategy. The next steps for this work are to publish an SOA research agenda with the results of our research. A longer-term goal for this work is the establishment of an SOA research community of interest to continue evolving the agenda as well as exploring the identified research topics.

REFERENCES

URLs are valid as of the publication date of this document.

[Abascal 2006]
Abascal, J., Arrue, M., Fajardo, I., & Garay, N. "An Expert-Based Usability Evaluation of the EvalAccess Web Service." *HCI-Related Papers of Interacción 2004*. Springer-Verlag, 2006.

[Afshar 2007]
Afshar M. & Moreland, B. *Keys to Successful Governance with SOA*. http://www.ebizq.net/topics/soa/features/7680.html

[Allen 2006]
Allen, P. *Service Orientation, Winning Strategies and Best Practices*. Cambridge University Press, 2006.

[Apache 2005]
The Apache Software Foundation. *Web Services - Axis*. http://ws.apache.org/axis/ (2005)

[Balzer 2004]
Balzer, Y. *Improve your SOA Project Plans*. http://www.ibm.com/developerworks/library/ws-improvesoa/ (2004)

[Bieberstein 2005]
Bieberstein, N., Bose, S., Walker, L., & Lynch, A. "Impact of Service-Oriented Architecture on Enterprise Systems, Organizational Structures, and Individuals." *IBM Systems Journal* (December 2005). http://www.highbeam.com/doc/1G1-140141564.html (2005)

[Bieberstein 2006]
Bieberstein, N., et al. *Service-Oriented Architecture Compass - Business Value, Planning and Enterprise Roadmap*. Pearson, 2006.

[Borck 2006]
Borck, J. "Planning an SOA: Gathering Around the Drawing Board." *InfoWorld* (May 2006). http://www.infoworld.com/article/06/05/08/77665_19FEsoalife2_1.html?s=feature (2006)

[Brandner 2004]
Brandner, M., Craes, M., Oellermann, F., & Zimmermann, O. "Web Services-Oriented Architecture in Production in the Finance Industry." *Informatik-Spektrum 27*, 2 (2004): 136–145.

[Brown 2006]
Brown W. & Cantor, M. *SOA Governance: How to Oversee Successful Implementation through Proven Best Practices and Methods* (IBM White Paper). ftp://ftp.software.ibm.com/software/rational/web/whitepapers/10706900_SOA_gov_model_app_v1f.pdf (2006)

[CBDi 2006]
CBDi. "Service Level Agreements: Best Practice Report." *CBDI Journal* (December 2006). http://www.cbdiforum.com/report_summary.php3?page=/secure/interact/2006-12/service_level_agreements.php&area=silver (2006)

[Cetin 2007]
Cetin, S., Altintas, N. I., Oguztuzun, H., Dogru, A.H., Tufekci, O., & Suloglu, S. "A Mashup-Based Strategy for Migration to Service-Oriented Computing," 169–172. *Proceedings of the IEEE International Conference on Pervasive Services*, Istanbul, Turkey, July 15–20, 2007. Digital Object Identifier 10.1109/PERSER.2007.4283910 (2007)

[Chawla 2007]
Chawla, M. & Peddinti, V. "Exposing SOA Enabled C Apps as Web Services." *SOA World Magazine* (February 2007). http://webservices.sys-con.com/read/314105.htm (2007)

[Chesbrough 2006]
Chesbrough, H. & Spohrer, J. "A Research Manifesto for Services Science." *Communications of the ACM 49*, 7 (July 2006): 35–40.

[Constantine 2000]
Constantine, L. "What Do Users Want? Engineering Usability into Software." http://www.foruse.com/articles/whatusers.pdf (2000)

[Droms 2002]
Droms, R. & Lemon, T. *The DHCP Handbook*. Sams Publishing. 2002.

[Eclipse 2008]
The Eclipse Foundation. *Web Tools Platform (WTP) Project*. http://www.eclipse.org/webtools/ (2008)

[Fitzgerald 2006]
Fitzgerald, B. & Olsson C. M. (eds.). "The Software and Services Challenge." Contribution to the Preparation of the Technology Pillar on *Software, Grids, Security and Dependability* in the European Union 7th Framework Programme. ftp://ftp.cordis.europa.eu/pub/ist/docs/directorate_d/st-ds/fp7-report_en.pdf (2006)

[Gold-Bernstein 2006]
Gold-Bernstein, B. & So, G. *Integration and SOA: Concepts, Technologies and Best Practices*. http://www.ebizq.net/webinars/7085.html (July 26, 2006)

[Hai 2006]
Hai, L., Li, Q., & Gu, N. "Quantifying Contexts for User-Centered Web Service Discovery," 399–404. *Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC 06)*. Hong Kong, October 16–20, 2006. Washington, DC (USA): IEEE Computer Society, 2006.

[High 2005]
High, R., Kinder, S., & Graham, S. *IBM's SOA Foundation: An Architectural Introduction and Overview*. http://download.boulder.ibm.com/ibmdl/pub/software/dw/webservices/ws-soa-whitepaper.pdf (November 2005)

[Horn 2005]
Horn, P. "The New Discipline of Services Science." *Business Week* (January 2005). http://www.businessweek.com/technology/content/jan2005/tc20050121_8020.htm

[Houlding 2007]
Houlding, D. "From SOA to SaaS: The Driving Forces behind Today's Software Architectures." *Dr. Dobbs Portal* (May 2007). http://www.ddj.com/architect/197700752/

[IBM 2004a]
IBM Research. *Services Science: A New Academic Discipline?* http://www.almaden.ibm.com/asr/SSME/facsummit.pdf (2004)

[IBM 2005]
IBM Business Consulting Services. *IBM Service-Oriented Modeling and Architecture*. http://www-935.ibm.com/services/us/gbs/bus/pdf/g510-5060-ibm-service-oriented-modeling-arch.pdf (2005)

[IBM 2007a]
IBM Corporation. *Web Service Level Agreements (WSLA) Project*. http://www.research.ibm.com/wsla/ (2007)

[IBM 2008]
IBM Corporation. *Services Sciences, Management and Engineering*. https://www.research.ibm.com/ssme/ (2008)

[InfoWorld 2007]
InfoWorld. *InfoWorld Research Report: Service Oriented Architecture(SOA), April 2007*. http://www.s2.com.br/s2arquivos/403/multimidia/197Multi.pdf (2007)

[Jini 2007]
Jini.org. *The Community Resource for Jini Technology*. http://www.jini.org/ (2007)

[Kajko-Mattsson 2007]
Kajko-Mattsson, Mira, Lewis, Grace, & Smith, Dennis. "A Framework for Roles for Development, Evolution and Maintenance of SOA-Based Systems," 7. *Proceedings of the International Workshop on Systems Development in SOA Environments (SDSOA 2007) at the International Conference on Software Engineering (ICSE 2007)*. Minneapolis, MN (USA), May 20–26, 2007. Digital Object Identifier 10.1109/SDSOA.2007.1

[Kajko-Mattsson 2008]
Kajko-Mattsson, Mira, Lewis, Grace, & Smith, Dennis. "Evolution and Maintenance of SOA-Based Systems at SAS." *Proceedings of the 41st Hawaii International Conference on System Sciences (HICSS-41)*. Waikoloa, Big Island, Hawaii (USA), January 7–10, 2008. IEEE Computer Society, 2008. http://csdl2.computer.org/comp/proceedings/hicss/2008/3075/00/30750119.pdf

[Kempf 1999]
Kempf, J. & St. Pierre, P. *Service Location Protocol for Enterprise Networks: Implementing and Deploying a Dynamic Service Finder*. John Wiley & Sons. 1999.

[Küster 2007]
Küster, U. & König-Ries, B. "Supporting Dynamics in Service Descriptions—The Key to Automatic Service Usage," 220–232. *Proceedings of Fifth International Conference on Service-Oriented Computing (ICSOC 2007)* (Lecture Notes in Computer Science, 4749). Vienna, Austria, September 17–20, 2007. Springer, 2007.

[Lewis 2007]
Lewis, G., Morris, E., Simanta, S., Smith, D., & Wrage, L. "SMART: Analyzing the Reuse Potential of Legacy Components in a Service- Oriented Architecture Environment." *Proceedings of the 2007 AIAA Infotech@Aerospace Conference.* Rohnert Park, CA (USA), May, 2007. AIAA, 2007.

[Linthicum 2006]
Linthicum, D. "When Building a SOA, How Do You Know When You're Done?" *InfoWorld* (June 2006). http://weblog.infoworld.com/realworldsoa/archives/2006/06/when_building_a.html (2006)

[Manes 2007]
Manes, A. T. *SOA Governance Infrastructure*. Burton Group, 2007.

[Marks 2006]
Marks, E., Bell, M. *Service Oriented Architecture: A Planning and Implementation Guide for Business and Technology*. John Wiley & Sons, 2006.

[Martens 2003]
Martens, A. "Usability of Web Services," 182–190. *Proceedings of the Fourth International Conference on Web Information Systems Engineering Workshops (WISEW'03)*. Rome, Italy, December 13, 2003. John Wiley & Sons, 2003.

[McClure 2006]
McClure, D. *Are IT Organizational Structures a Barrier to Business Service Management Success?* http://dougmcclure.net/blog/2006/11/are-it-organizational-structures-a-barrier-to-business-service-management-success/ (2006)

[Nielsen 1993]
Nielsen, J. *Usability Engineering*. Morgan Kaufmann, 1993

[OASIS 2007]
OASIS. *Web Services Context Specification (WS-Context) Version 1.0*. http://docs.oasis-open.org/ws-caf/ws-context/v1.0/wsctx.html (2007)

[Pai 2007]
Pai, Y. *Ideal IT Organization for Business Agility*.
http://soablueprint.com/yahoo_site_admin/assets/docs/SOA_Ideal_Organization.9153115.pdf (2007)

[Pujari 2004]
Pujari, D. "Self-Service with a Smile? Self-Service Technology (SST) Encounters among Canadian Business-to-Business." *International Journal of Service Industry Management 15*, 2 (2004): 200–219.

[Rodriguez 2005]
Rodriguez, J. *New Rules Govern SOA Lifecycle*.
http://www.looselycoupled.com/opinion/2005/rodri-rules-gov0701.html (July 2005)

[Shodjai 2008]
Shodjai, P. *Web Services and the Microsoft Platform*.
http://msdn2.microsoft.com/en-us/library/aa480728.aspx (2008)

[Sneed 2006]
Sneed, H. "Integrating legacy Software into a Service Oriented Architecture," 3–14. *Proceedings of the 10th European Conference on Software Maintenance (CSMR 2006)*. Bari, Italy, March 22–24 2006. IEEE Computer Society Press, 2006. http://ieeexplore.ieee.org/iel5/10671/33675/01602353.pdf?tp=&isnumber=&arnumber=1602353

[Sneed 2007]
Sneed, H. "Migrating to Web Services: A Research Framework." *Proceedings of the International Workshop on SOA Maintenance Evolution (SOAM 2007), 11th European Conference on Software Maintenance and Reengineering (CSMR 2007)*, Amsterdam, the Netherlands, March 20–23, 2007. http://www.cs.vu.nl/csmr2007/workshops/4-%20SneedSOAPaper.pdf

[Spohrer 2006]
Spohrer, J. & Maglio, P. *The Emergence of Service Science: Toward Systematic Service Innovations to Accelerate Co-creation of Value.* http://www.almaden.ibm.com/asr/SSME/jspm.pdf (2006)

[Tilley 2004]
Tilley, S., Gerdes, J., Hamilton, T., Huang, S., Müller, H. A., Smith, D., & Wong, K. "On the Business Value and Technical Challenges of Adopting Web Services." *Journal of Software Maintenance and Evolution: Research and Practice 16*, 1–2 (2004): 31–50.

[UDDI 2008]
UDDI XML.org. *Welcome to UDDI XML.org*. http://uddi.xml.org/ (2008)

[UPnP 2008]
UPnP Forum. *Welcome to the UPnP Forum*. http://www.upnp.org/ (2008)

[Veryard 2004]
Veryard, R. *The SOA LifeCycle*. CBDi, August 2004.

[W3C 2004a]
W3C. *OWL-S: Semantic Markup for Web Services*. http://www.w3.org/Submission/OWL-S/ (November 2004)

[W3C 2005a]
W3C. *Semantic Web Services Framework (SWSF) Overview*. http://www.w3.org/Submission/SWSF/ (September 2005)

[W3C 2005b]
W3C. *Web Service Semantics - WSDL-S*. http://www.w3.org/Submission/WSDL-S/ (November 2005)

[W3C 2007]
W3C. *Semantic Annotations for WSDL and XML Schema*. http://www.w3.org/TR/sawsdl/ (August 2007)

[Windley 2006]
Windley, P. "SOA Governance: Rules of the Game." *InfoWorld* (January 2006). http://www.infoworld.com/pdf/special_report/2006/04SRsoagov.pdf

[Winter 2007]
Winter, A. & Ziemann, J. "Model-Based Migration to Service-Oriented Architectures." *Proceedings of the International Workshop on SOA Maintenance Evolution (SOAM 2007), 11th European Conference on Software Maintenance and Reengineering (CSMR 2007)*, Amsterdam, the Netherlands, March 20–23, 2007. http://www.cs.vu.nl/csmr2007/workshops/2-%20winterziemann.pdf

[Woolf 2005]
Woolf, B. *Streamline SOA Development Using Service Mocks*. http://www-128.ibm.com/developerworks/library/ws-mocks/ (2005)

[WSMO 2008]
WSMO. *Web Service Modeling Ontology*. http://www.wsmo.org (2008)

[X500 2008]
X500Standard.com. *Welcome*. http://www.x500standard.com/ (2008)

[Ziemann 2006]
Ziemann, J., Leyking, K., Kahl, T., & Werth, D. "SOA Development Based on Enterprise Models and Existing IT Systems." *Exploiting the Knowledge Economy: Issues, Applications and Case Studies* (P. Cunningham & M. Cunningham, eds.). IOS Press, 2006.

3 Towards Adaptive Service Engineering

Authors: Daniel Schneider,[6] Christian Bunse,[7] and Klaus Schmid[8]

Keywords: system adaptivity, context-awareness, ambient intelligence, ubiquitous, pervasive, service-based, service-orientation

Acknowledgement: The creation of this paper was partly supported by AmbiComp (01ISF05A) project, funded by the German Federal Ministry of Education and Research (BMBF).

Abstract: Technical systems are increasingly becoming an imminent part of human life. A growing trend is that systems are embedded in technical devices and working continuously without human intervention. However, this implies that these systems run for a long time without human control. As the environment changes, the systems need to adapt themselves. One approach to address these challenges is the use of service-oriented development paradigms. This paper highlights the challenges and research issues in the context of engineering adaptable, service-oriented systems. Challenges are identified based on the development plane they might appear in (i.e., service-, application-, and infrastructure engineering). We finish by discussing in detail the expected benefits and open research issues.

3.1 INTRODUCTION

Software systems are becoming more and more an imminent part of human life. Especially the amount of software systems that are not directly visible and/or working (together) in the background is constantly increasing. This, in turn, has also increased the need for software systems that can adapt themselves at runtime (i.e., to react to varying resources, errors, or changing requirements).

A good example for such systems is the domain of ambient-intelligence (AmI) systems (e.g., assisted living, smart-home, etc.). In its vision, the AmI paradigm is related to ubiquitous computing systems as described by M. Weiser in 1993 [Weiser 1993], but puts a special emphasis on proactive and intelligent behaviour. Such systems have to react to varying resources (e.g., bandwidth, accessibility of servers, etc.) and errors. Users are demanding for systems that guarantee for a specific quality of Service (QoS), security, safety, and flexibility of platforms. In addition, such systems have to adapt to changing requirements, and heterogeneous infrastructures, while working autonomously. Additional challenges arise from the fact that AmI systems heavily rely on embedded devices with scarce resources (i.e. energy, memory, processing power).

The recent advent of the service-oriented-architectures (SOA) paradigm seems to be a promising step forward. The core idea behind SOA is to enable the dynamic selection, orchestration and ex-

[6] Fraunhofer Institute for Experimental Software Engineering, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany, Daniel.Schneider@iese.fraunhofer.de

[7] School of IT, International University in Germany, Campus 2, 76646 Bruchsal, Germany, Christian.Bunse@i-u.de

[8] Institute of Computer Science, University of Hildesheim, D-31141 Hildesheim, Germany, schmid@sse.uni-hildesheim.de

ecution of services (i.e., discretely defined sets business or technical functionality) to achieve a specific result for the service user – who might be a real person as well as another service. Thus, the application of SOA promises increased flexibility and maintainability, independency of technology and functionality, improved reuse, and ease of communication. However, technologies and methods are needed to use SOA effectively for the development of adaptive systems.

When applying SOA principles during software development it appears that frequently a number of candidate services or components are available that might or might not be used during system composition. Unfortunately, these often do not match perfectly to the given requirements and have to be matched as "best fit" based on their different properties. The relevant criteria usually also include extra-functional properties as, for instance, a specific quality of service (QoS). Obviously, one of the major challenges is to answer the question which services should be used in order to obtain a user-acceptable behaviour – and which should be avoided. Moreover, the compliance with the negotiated contracts has to be monitored over the execution time. Based on changes in user expectations, resource situations, service landscape or environmental conditions it may be necessary to adapt specific services (or compositions thereof). To realize such adaptive behaviour, the corresponding adaptation requirements need to be taken into account throughout development and must be realized by adequate runtime mechanisms in the architecture.

The envisioned solution towards this problem, as highlighted in this paper, is the creation of a development framework that facilitates the understanding and application of the entire spectrum of adaptive mechanisms in engineered systems. Thus, the framework has to cover the engineering of services, the composition of services into new systems (i.e., application engineering), and the infrastructure (e.g., protocols, message exchange, etc.).

The remainder of this paper is structured as follows. Subsection 3.2 discusses the issues of integrating adaptation into the service-oriented paradigm by examining various forms of adaptation and a concrete application scenario. Subsection 3.3, examines the areas of service-, application, and infrastructure engineering, and discusses the most prominent challenges in the development of adaptive service-based systems for each area. Finally, Subsection 3.4 provides a short summary and an outlook towards upcoming research.

3.2 ADAPTATION IN SERVICE-BASED SYSTEMS

While adaptation is certainly not the only challenge in service-based systems, it is nevertheless an important one. In particular, the inherently dynamic nature of ambient systems strongly requires the existence of adaptation mechanisms in order to address issues of appearing and disappearing services, as well as to react to problems with quality levels of services.

3.2.1 Background

Approaches to software adaptation usually differ with regard to when the adaptation occurs and to how the adaptation is performed. As for the former, adaptation can occur at development time, at compile/linking time, at load time and at runtime. And as for the latter, adaptation can be accomplished by a change in algorithms/behaviour, by a change of parameters or by a change of the software composition. As an example consider the use of compositional adaptation in different approaches for software adaptation at different points in time: At development time compositional adaptation is employed in the course of Software Product Lines (SPL) [Clements 2001] or Component-Based Software Engineering (CBSE) [Heineman 2001] with custom-off-the-shelf (COTS)

components. At compile time, compositional adaption occurs as an ingredient of Aspect Oriented Programming (AOP) [Kiczales 1997]. Load time composition is, for instance, performed by Java Virtual Machines. Dynamic composition at runtime is a constituent of Service Oriented Architectures (SOA).

Of course, in the context of SOA, different types of adaptation at different points in time could be beneficial (consider a combination of SOA and SPL e.g.). However, independent from the actual "how" (i.e. adaptation of composition, behaviour or parameters), the most significant step regarding increasing dynamicity, and unfortunately also regarding increasing complexity of the realization, is when adaptation is taken into runtime. Runtime adaptation implies the existence of mechanisms that acquire and assess the current situation and corresponding means to adapt the system accordingly. With that being an integral part of adaptive service-based systems we will further elaborate on this aspect in the remainder of this section.

3.2.2 Dynamic Adaptation in Service-Based Systems

In the following we describe a typical application scenario which illustrates the types of dynamic adaptation that could occur (i.e., in an AmI system): Imagine the user wants to have a video conference. The application knows how to map the abstract service characterization to ranges of acceptable qualities (i.e. non-functional or QoS properties) and specific functional parameters of underlying services. This mapping may also be influenced by current context information as, for instance, the service user and his location. Regarding this concrete example, let us assume the application requires a video service and an audio service at a certain user specific quality. The information on the required service is submitted to the infrastructure as a look-up request. The infrastructure has then to match the request to corresponding services. Services in turn describe the mapping of their provided functionalities and corresponding non-functional properties to low-level resources. This information is also used by the infrastructure for eventually selecting a service or a service composite (i.e. service aggregate consisting of many other services; in this case a "video conference" service consisting of a video service and an audio service) that suits the user needs as well as the resource situation. Subsequently, a corresponding service contract can be established. Such service contracts must be managed actively to ensure contract compliance over the service execution time.

Dynamic composition obviously plays a central role in adaptive service-based systems. It is required to tackle the dynamic orchestration of services based on their specifications, the current resource situation and available context information. However, as hinted at the end of our example, another important aspect is the maintenance of contract conformance over the service execution time. Contract violations must be recognized to trigger appropriate counter measures as, for instance, renegotiation of contracts or even dynamic recomposition of services. To this end, it may not be adequate to use compositional adaptation alone considering fast changing non-functional properties, resources and context as triggers for dynamic adaptation. This would either lead to a rather coarse grained adaptation behaviour which adapts only in some rare occasions or to a rather inefficient treatment with a huge overhead due to steady recompositions. A more viable approach would be to define contracts with ranges of acceptable values of properties and to establish adaptation mechanisms that maintain such properties within the predefined ranges. Such adaptation mechanisms must be relatively "light weight" compared to compositional adaptation since they ought to operate on a much shorter timescale. Renegotiation of contracts or even recomposition of service composites would then only occur when it is no longer possible to main-

tain the service properties within the predefined ranges. In fact, this idea of having different types of adaptation mechanisms, of different weight, working on different timescales, is not new for other domains. In the domain of QoS-aware communication, for instance, it is common to distinguish between QoS-control and QoS-management mechanisms [Aurrecoechea 1998].

These considerations naturally strongly influence the challenges in the development of such systems as presented in Section 3.3.

3.3 CHALLENGES IN THE DEVELOPMENT OF ADAPTIVE SERVICE-BASED SYSTEMS

In the development of adaptive service-based systems we can distinguish three major activities: *service engineerin*g, *application engineering* and *infrastructure engineering*. In the following we will further elaborate on those three activities and discuss the specific challenges therein.

3.3.1 Service Engineering

Services that are provided independently to concrete applications by providers must find their market, i.e., specific applications that are relevant to them. As a consequence, we need to systematically identify specifications of services that will be market-relevant. During runtime, it must be possible to identify these services appropriately, thus service specifications are required as a basis for lookup, negotiation and composition of services. As illustrated in Section 2, these specifications must describe both, functional and non-functional properties of the service. As for the functional specification, a widely recognized specification technique is the *component interface definition language* (CIDL) [OMG 1999]. A good survey on current work in specification techniques for non-functional properties can be found in [Jin 2004]. In the web-service community the *web services description language* (WSDL) [W3C 2001] is widely used. In order to automate the identification of the service capabilities, matching of specifications and composition of services, a semantic service specification would be required. This, however, also implies the need for corresponding infrastructure mechanisms (i.e. semantic matchmaking).

In addition to the typical issues to be addressed by a method (i.e., defining what, when, and how to do) a service engineering methodology should specifically address adaptability. This also includes support for the integration of adaptation mechanisms (graceful degradation and scaling e.g.) into services like introduced in Subsection 3.2.

3.3.2 Application Engineering

Application engineering is the development of new, possibly adaptive systems while integrating pre-existing services. Interestingly, the problems and challenges are closely related to those of the component world that aim at building systems with COTS components. However, service-orientation adds a new level of difficulty by aiming at adaptation and “open” connectivity. In general, according to [Bunse 2006] the expectation of building systems from prefabricated parts is the belief that prefabricated parts will provide good reliability, will help to save time and cost, and will follow the market trend (i.e., the evolution of the selected part will follow the market trend). Although, there is no direct influence on evolution, users still trust that the vendor will update the part according to market needs. Despite these hopes of simply engineering new systems from prefabricated parts, components, or services, several risks must be considered. On the one hand developers must take possible adaptations into account. Furthermore, it is necessary to monitor the "wider" environment as well as own capabilities (e.g., location, user capabilities, results of system

itself, etc.) to be able to react quickly to changing factors. The environment and its varying context factors must be examined in order to define the level of adaptability the system needs. In addition, it is also important to be aware of user needs since reactions/adaptations must be in the interest of the user.

Application engineering methods should explicitly support the acquisition of relevant information and the flexible specification of service-based systems. In particular, this requires the specification of adaptation and reasoning techniques. Concerning the first challenge possible candidates of such techniques are pre-defined architectures or architecture-patterns offering solutions to recurring problems in the development of service-based systems. Concerning the second, techniques are needed to "reason" (pre-coded, or explicit reasoning) about potential adaptations. This also requires considering issues of scalability.

3.3.3 Infrastructure Engineering

Infrastructure engineering supports the development of specific, yet open, platforms for adaptive services. Such platforms are specific with respect to their capabilities and they must be open towards services and applications that might not be known at development time. The infrastructure must provide standard mechanisms that realize the central functionalities of a service oriented system. As illustrated in Section 2, such mechanisms include service brokers, facilities for (semantic) lookup, specification matching and service composition, context management and specific QoS functionalities.

A corresponding engineering framework might incorporate standard (generic) architectures, standard mechanisms (realizing SOA functionalities, QoS and adaptation), corresponding patterns and guidelines to build concrete systems based on the constituents named before.

3.4 SUMMARY/OUTLOOK

This paper gave an overview on adaptive service engineering, its challenges, expected benefits and open research issues. We based our discussion on a distinction of the three development planes of adaptive service-based systems: Service Engineering, Application Engineering and Infrastructure Engineering. Concerning service engineering we see challenges in the specification, covering functional and non-functional properties, as well as in the identification of relevant services at development and run-time. This also includes methodological support for the integration of adaptation mechanisms.

Concerning application engineering we see challenges that are comparable to those of component-based development wrt. to quality, risks, etc. Furthermore, it is necessary to examine the environment and its varying context factors for defining the level of adaptability a system needs. Therefore, the acquisition of relevant information and the flexible specification of service-based systems is a major challenge.

Concerning infrastructure engineering we need standard mechanisms, including service brokers, facilities for (semantic) lookup, specification matching and service composition, context management and specific QoS functionalities in order to realize the central functionalities of a service oriented system.

As a consequence, we want to build a framework that provides means (methods, tools, etc.) for specifying and modelling adaptive services, for integrating mechanisms that ensure adaptation at development and run-time, and means to handle infrastructure needs.

More precisely, we want to develop approaches and tools, covering the explicit description of functional and non-functional service properties that can be provided or are required. Moreover, based on those specifications, application developers should be supported concerning the identification of the "best-fit" services according to their needs. As for the infrastructure, there needs to be support with regard to the integration of mechanisms that cover the central SOA functionalities, mechanisms that monitor and manage contract compliance at runtime, and mechanisms that realize context management. To this end, we want to propose generic architectures, standard mechanism, design patterns and guidelines for the instantiation of concrete architecture.

REFERENCES

URLs are valid as of the publication date of this document.

[Aurrecoechea 1998]
Aurrecoechea, C., Campbell, A., & Hauw, L. "A Survey of QoS Architectures." *ACM/Springer Verlag Multimedia Systems Journal 6*, 3 (May 1998): 138–151.

[Bunse 2006]
Bunse, C., Conradi, R., Li, J., Morisio, M., Slyngstad, O.P.N., & Torchiano, M. "An Empirical Study on the Decision Making Process in Off-The-Shelf Component Based Development," 897–900. *Proceedings of the Emerging Results Track at the 28th International Conference on Software Engineering (ICSE 2006)* (Leon J. Osterweil, H., Dieter Rombach, and Mary Lou Soffa, eds.), May 2006, Shanghai, P.R. China. ACM, 2006.

[Clements 2001]
Clements, P. C. & Northrop, L. *Software Product Lines: Practices and Patterns.* Addison-Wesley, 2001.

[Heineman 2001]
Heineman, G. T. & Councill, W. T. *Component-Based Software Engineering: Putting the Pieces Together*. Addison-Wesley Professional, 2001.

[Jin 2004]
Jin, J. & Nahrstedt, K. "QoS Specification Languages for Distributed Multimedia Applications: A Survey and Taxonomy." *IEEE MultiMedia, 11*, 3 (2004): 74−87.

[Kiczales 1997]
Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J. M., & Irwin, J. "Aspect-Oriented Programming," 220–242. *Proceedings of the 11th European Conference on Object-Oriented Programming (ECOOP '97) (Lecture Notes in Computer Science, 1241)* Jyväskylä, Finland, July 9–13, 1997. Springer, 1997.

[OMG 1999]
Object Management Group. *CORBA Component Model Joint Revised Submission OMG Document orbos/99-07-01* OMG, July 1999.

[W3C 2001]
W3C. *Web Services Description Language (WSDL) 1.1* (W3C Note 15 March 2001). http://www.w3.org/TR/wsdl

[Weiser 1993]
Weiser, M. "Some Computer Science Issues in Ubiquitous Computing*." Communications of the ACM 36*, 7 (1993): 75−84.

4 Bridging the Gap Between Object-Oriented Programming and Service-Oriented Computing

Authors: Sven De Labey,[9] Marko van Dooren, and Eric Steegmans[10]

Abstract: Object-oriented programming languages deliver the main technology for implementing enterprise systems, but they are losing pace with the rapidly evolving paradigm of Service-Oriented Computing. This is mainly due to inadequate support for dealing with service volatility and distribution issues. In this paper, we focus on the growing distance between Java and Service Oriented Computing. First, we discuss the disadvantages of the current Java programming model in the context of developing SOA applications. Then, we provide a basis for bridging the gap between both programming paradigms.

4.1 INTRODUCTION

The emergence of Service Oriented Architectures (SOA) imposes new challenges on software development [Papazoglou 2003]. These challenges originate from (1) the distributed nature and (2) the inherent volatility of remote services. Being distributed, remote services are confronted with availability problems due to network errors and server outages. Being volatile, service applications must cope with newly joining services, service migration, incompatible service updates and service deprecation. Additionally, customers require services to be tailored to their own personal needs, thus creating a need for dynamic service selection and Quality of Service (QoS) negotiation.

Platforms such as Java Enterprise Edition 5.0 [JCP 2006] offer the main technology for implementing enterprise systems, but their programming models only support static service binding based on URLs hardcoded in source code annotations and deployment descriptors. Consequently, support for dynamic service selection and failover must be implemented over and over again, making Java less appealing for the implementation of applications running on dynamic service architectures.

In this paper, we show that object-oriented programming languages need specialized language constructs for dealing with the new challenges introduced by SOAs. This language extension improves other object-oriented solutions in two ways. First, it allows us to build highly available SOA applications by providing language concepts for failover. Second, the extension deals with the volatile nature of services by supporting late service binding, QoS negotiation, and optimal service selection.

This paper is structured as follows. Subsection 4.2 shows how Java fails to provide adequate support for SOA applications and Subsection 4.3 shows how these problems can be tackled by introducing new language constructs. We present related work in Subsection 4.4 and we conclude in Subsection 4.5.

[9] Sven De Labey is funded by the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

[10] K.U.Leuven, Dept. of Computer Science, B-3000 Leuven, Belgium {svendl, marko, eric}@cs.kuleuven.be.

4.2 THE JAVA PROGRAMMING MODEL FOR SERVICE APPLICATIONS

From the release of the Java API for XML-based Web Services 2.0 (JAX-WS [JCP 2005]), the Java Enterprise Edition programming model [JCP 2006] includes an annotation, @WebServiceRef, to define a web service reference. The information found in this annotation typically consists of (1) the location of the remote WSDL file and (2) the type of the service interface, as shown in the code sample below. The runtime system uses this information to initialize the field with a reference to the remote service endpoint.

```
@WebServiceRef{
wsdlLocation="www.wsdlurl.com"
type=T.class}
T myService; //initialized by container \
```

From the viewpoint of service-oriented computing, this level of support is problematic for a number of reasons:

1. Static Service Binding. The use of wsdlLocation attributes hardwires service references to physical web service locations, thus requiring code revisions to handle service migration and service deprecation. But this is not manageable because the annotations to be updated may be scattered throughout the entire application. An alternative approach could be to externalize service addresses in a deployment descriptor [JCP 2006], but even then, the dynamic nature of web services causes (1) frequent and complex revisions of a verbose XML file and (2) application redeployment.
2. Flexibility. There is no support for dynamically selecting and injecting the optimal service from a set of competing services because each instance variable is bound to exactly one service endpoint. By forcing customers with different QoS requirements to use the same service, however, the flexibility of the entire application is severely limited.
3. Distribution. @WebServiceRef annotations make it impossible to reroute requests to backup endpoints. If a service crashes, the client must either wait until that service recovers, or wait until someone updates the annotation and redeploys the application. Both solutions are unacceptable.
4. Reusability. Exporting the application to a new service architecture forces developers to rewire every @WebServiceRef annotation to the services in that new environment. This is not manageable because the annotations to be updated are scattered throughout the entire application.
5. Program Comprehension. Java makes the programmer responsible for implementing service selection and failover. These algorithms cooperate tightly with the business logic, which obstructs a clean abstraction into reusable libraries or aspects. But the occurrence of these algorithms in the business logic decreases both the readability and the modularity of the source code, which in turn increases maintenance costs as well as the probability to introduce bugs in newer versions of the software.
6. Web Service Sessions. Complex business operations often comprise multiple client-service interactions, grouped into a session. Java offers no support for demarcating such sessions, thus forcing programmers to devise their own transaction semantics for each business operation.

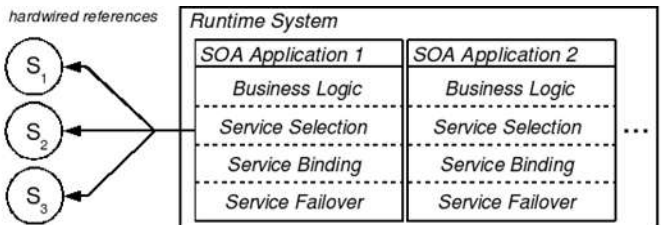

Figure 4-1: Monolithic Applications with Code Duplication and Hardwired References

Towards Language Support for Agile Service Applications

We propose to extend the Java programming model in two ways. First, we introduce *declarative language concepts* that offer support for dynamic service selection, transparent failover, and the definition of QoS constraints. Second, we make the compiler responsible for injecting algorithms that handle these non-functional requirements based on information gathered from our new language constructs. This allows programmers to focus on the business logic, leaving all technical issues to the middleware, which allows for modular software development, as shown in Figure 4-2.

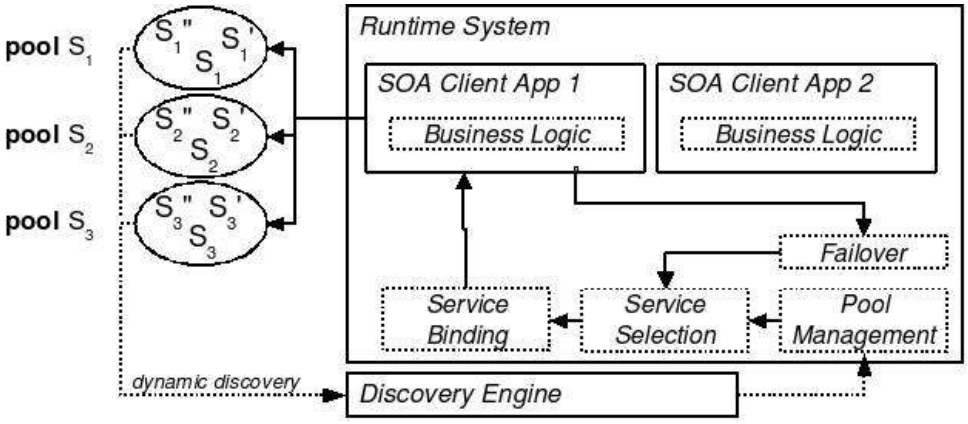

Figure 4-2: Modular Applications with Service Pools and Increased Runtime Support

4.3 LANGUAGE CONSTRUCTS FOR SOA APPLICATIONS

We define two constructs for dealing with the problems mentioned in Subsection 4.2. First, *service pools* allow for late service binding and transparent failover. Second, *declarative operations* enable developers to specify QoS constraints and quality attributes for optimizing and personalizing service selection.

4.3.1 Service Pools as a Type System Extension

In Subsection 4.2, we have shown that transparent failover and dynamic service selection are not supported in Java because the container can only inject a single reference per annotated instance field. We solve this problem by introducing the concept of a pool variable. Instead of referring to a hardwired service URL, such pool variables point to a collection of interchangeable service endpoints. The runtime transparently injects a service from this collection, and in case of a failure, reinvokes the operation on another pool member (typically a replica of the failing service) without a need for user intervention.

Pool variables differ from normal Java variables in that they have a type qualifier, **pool**, to indicate that the compiler must insert algorithms for service selection and transparent failover. But there is no syntactic difference between invoking operations on pools and invoking operations on normal fields: the logical view of a service pool is *a single remote reference*. The following code, for instance, declares a pool variable and invokes a fault tolerant operation on a service selected from the corresponding pool fsP:

```
pool FlightService fsP;   //transparent pool initialization
fsP.bookFlight();//transparent failover
```

4.3.2 Pool Initialization

Pool references should not be initialized by application programmers since this would bring back the problems of hardwired service references. Instead, pools are transparently initialized by the pool manager, a new service of the runtime system (see Figure 4-2). This manager scans through the service path [De Labey 2006] to find appropriate services. Such a service path is similar to the Java classpath, but it consists of service URLs instead of class locations. Another way of adding service endpoints is by relying on external discovery mechanisms. Pool initialization is discussed in more detail in our technical report [De Labey 2006].

4.3.3 Enforcing QoS Constraints on Service Pools

Often, programmers need to limit pool membership to those services that comply with a number of QoS constraints. By lacking runtime support for QoS negotiation, Java forces programmers to write code for iterating through a list of candidates, querying them for QoS properties, and retaining those services that comply with the imposed QoS constraints. We avoid this repetitive work by introducing a declarative operation, **where**, that accepts the QoS constraints as a boolean condition (this is shown in the left part of Figure 4-3). The runtime system is responsible for constraining the pool to those services that comply with this condition. Creating a pool of services that provide a flight from locX to loxY, for instance, can be written as:

```
pool FlightService fsSeq where fsSeq.offersRoute(locX,locY);
```

4.3.4 Optimal Service Selection based on Preferences

The pool qualifier is used to denote a collection of *equivalent services* and allows to view a set of replicated services as a single service endpoint. Often, however, some services in the pool have more interesting properties than others, thus breaking our equivalence assumption, making the use of the pool qualifier less appropriate. Therefore, we introduce a subtype of the pool qualifier, **sequence**, to indicate the existence of these *user preferences* in a service pool. Preferences are specified using the **orderby** operation (this is shown in the right part of Figure 4-3). For example, customers willing to minimize their traveling costs, may induce an order on the FlightService pool by creating a sequence with the ticket price as a sorting attribute:

```
sequence FlightService fsSeq orderby fsSeq.getTicketPrice();
```

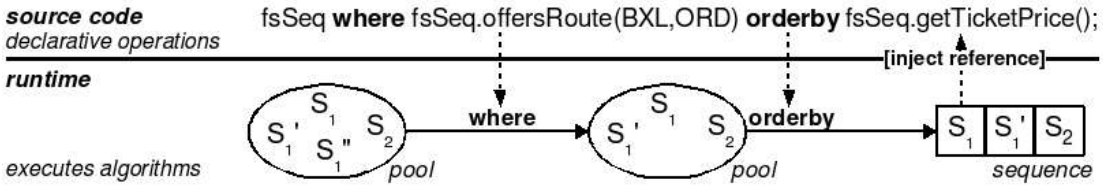

Figure 4-3: Using Declarative Operations to Define QoS Constraints and Preferences

4.3.5 Pool Sessions Support Ad Hoc Web Service Transactions

The execution of a complex business operation often comprises multiple interactions between the client and a web service. This set of method invocations is called a session, and for consistency reasons, it is crucial that the pool variable is bound to the *same service for the entire session*. Therefore, we introduce **`session{. . . }`** blocks to combine related web service interactions into sessions. On entering a session block, the runtime injects a service into the pool variable, which is then locked until the session ends successfully, or until the injected service becomes unreachable. In the latter case, the runtime injects another pool member, and restarts the entire session. Thus, session-scoped locks enable atomic, fault-tolerant interactions with remote services.

4.4 RELATED WORK

Language constructs for fault tolerance were proposed by Cardelli in [Cardelli 1999]. Related concepts are available in WebOz [Hadim 2000], WebL [Kistler 1998], XL [Florescu 2003], and Aries [Pereira 2004]. Our approach extends this proposal with declarative operations for optimizing service selection.

ActiveXML and BPEL [IBM 2003a] suffer from hardwired service references, but this problem is solved by introducing implicit service invocation in [Benbernou 2005]. This approach resembles our proposal, but developers are still forced to pollute their business logic with code for handling availability problems.

A lot of solutions based on Aspect-Oriented Programming have been proposed to deal with cross-cutting concerns such as security, logging and fault tolerance. The proposal that most resembles our approach is the Web Services Management Layer (WSML) [Verheecke 2004], which enables hotswapping of unreachable services. We provide transparent failover in a similar way and add support for fine-tuning service selection (**`where`** and **`orderby`**).

4.5 CONCLUSION

The Java programming model relies on hardwired service URLs and offers no support for important SOA challenges such as dynamic service selection and binding, transparent failover, QoS negotiation, and web service sessions. Forcing developers to implement these algorithms decreases the readability and maintainability of the source code, at the same time limiting reusability and flexibility of the SOA application.

We argue that an extension of Java is necessary to deal with these new challenges and equip developers with *type qualifiers*, *declarative operations*, and *session blocks*. These concepts foster the development of modular applications that can handle dynamic changes in the service architecture without requiring programmer intervention.

REFERENCES

URLs are valid as of the publication date of this document.

[Benbernou 2005]
Benbernou, S., et al. "Implicit Service Calls in ActiveXML Through OWL-S," 353–365. *Proceedings of the Third International Conference on Service Oriented Computing (ICSOC05) (Lecture Notes in Computer Science 3826)*. Amsterdam, The Netherlands, December 12–15, 2005. Springer, 2005.

[Cardelli 1999]
Cardelli, L. & Davies, R. "Service Combinators for Web Computing." *IEEE Transactions on Software Engineering 25*, 3 (1999): 309−316.

[De Labey 2006]
De Labey, S., van Dooren, M., Steegmans, E., & Service, J. *Towards Service-Oriented Programming in Java* (Technical Report CW451). http://www.cs.kuleuven.be/publicaties/reports/CW/2006/. (2006)

[Florescu 2003]
Florescu, D., Gruenhagen, A., & Kossmann, D. "XL: A Platform for Web Services." *Proceedings of the First Biennial Conf. on Innovative Data Systems Research.* Asilomar, CA (USA), January 2003. http://www.dbis.ethz.ch/research/publications/47.pdf

[Hadim 2000]
Hadim, M., et al. "Service Combinators for Web Computing in Distributed Oz," Volume IV. *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2000)*. Las Vegas, NV (USA) June 24–29, 2000. CSREA Press, 2000.

[IBM 2003a]
IBM Corporation. *BPEL4WS Specification v1.1*. http://www.ibm.com/developerworks. (2003)

[JCP 2005]
JCP. *JAX-WS 2.0 Specification*. http://www.jcp.org/en/jsr/detail?id=224 (2005)

[JCP 2006]
JCP. *Java EE 5.0 Specification*. http://www.jcp.org/en/jsr/detail?id=244 (2006)

[Kistler 1998]
Kistler, T. & Marais, H. "WebL - A Programming Language for the Web." *Computer Networks and ISDN Systems 30*, 1-7 (April 1998): 259−270.

[Papazoglou 2003]
Papazoglou, Mike. "SOC: Concepts, Characteristics and Directions," 3–12. *Proceedings of the Fourth International Conference on Web Information Systems Engineering Workshops (WISEW'03)*. Rome, Italy, December 13, 2003. John Wiley & Sons, 2003.

[Pereira 2004]
Pereira, F., Valente, M., et al. "Tactics for Remote Method Invocation." *Journal of Universal Computer Science 10*, 7 (2004): 824−842.

[Verheecke 2004]
Verheecke, B., et al. “AOP for Dynamic Configuration and Management of Web Services.” *International Journal on Web Services Research 1*, 3 (2004): 25−41.

5 Model Driven Development of Service Oriented Architectures—Transforming Business Logic into Service Infrastructures

Authors: Xabier Larrucea, Gorka Benguria, Stefan Schuster[11]

Keywords: SOA, MDA, Business Modelling, Transformation

Acknowledgement: This work was funded by the European Integrated Project ATHENA (IST-2003-507849) and the authors would like to thank the project partners for their feedback.

Abstract: Service Oriented Architecture (SOA) is a new paradigm for building loosely coupled systems and is considered particularly useful for integrating existing system components and applications. It furthermore facilitates the re-use of existing assets and the management of the inherent complexity of systems.

Business processes are used to define the business logic of an organisation. The practice of modelling these business processes is a means for coherently and consistently representing all elements involved in the operational processes of the organisation at conceptual level. But when using these business process models as requirements specification for the ICT infrastructure that is supposed to support them, the "Chinese whispers" effect comes into play. The different stakeholders interpret the information captured in these models from their specific viewpoint, which usually leads to misconceptions and loss of information.

This paper presents the experiences made in applying a Model Driven Service Oriented Architecture (MDSOA) framework as a potential solution for the current situation and discusses the shortcomings and challenges of this concept as of today.

5.1 INTRODUCTION

Due to phenomena like globalisation [Levitt 1983], outsourcing and ever faster changing market needs, to name only a few, the need to increase revenue figures by reducing production and maintenance costs [Ayad 2002] becomes quintessential for many organisations. In order to respond to this situation, the software industry applies several strategies to reduce the time needed to adapt the existing ICT (Information and Communication Technology) infrastructure to changing business needs, which range from methodological approaches [Griss 2002], such as agile techniques, to infrastructural approaches, such as service orchestration tools [Peltz 2003].

Service Oriented Architecture (SOA) [W3C 2004b] is a new paradigm for building loosely coupled systems and is considered particularly useful for integrating existing system components and applications. It furthermore facilitates the usage of existing assets (reuse) and the inherent complexity management of systems [Endrei 2004].

[11] European Software Institute, Parque Tecnológico # 204, E-48170 Zamudio, Spain
{Xabier.Larrucea,Gorka.Benguria,Stefan.Schuster}@esi.es

Business processes are used to define the business logic of an organisation and several languages, some of them under standardization, exist for defining them [IBM 2004b, Troux 2002]. The usage of business process models brings several advantages when an organisation is planning to change its actual structure to achieve greater efficiency. Business process models are capable of coherently and consistently representing all elements involved in the operational processes of the organisation at conceptual level, which helps users to get a common understanding of the process from different perspectives. Unfortunately, the transformation of business process models into an ICT infrastructure is in many cases not sufficiently formalized: When simply using process models as requirements specification for system development, the "Chinese whispers" effect comes into play. The different stakeholders, in this case, business expert and system developer, interpret the information captured in these models from their specific viewpoint, which usually leads to misconceptions and loss of information. The resulting information systems consequently lack of flexibility and traceability and the complexity of checking the consistency between the business and system layer increases.

As a result of this situation, three major problems can be summarized:

1. Organisations do not use a standard, unified and widely adopted business process definition language.
2. Information systems are implemented in a proprietary format, on a specific platform and their communications and connections are not easy to implement.
3. Information systems do not support the business processes. There is a gap between business process models and their information systems implementations.

The Model Driven Architecture (MDA) [OMG 2003] initiative is promoting the usage of models for describing, building and deploying system architectures. Applying an MDA vision to enterprise architectures as well as system architectures provides the means to build model based systems that avoid or reduce the loss of information while they increase the separation of concerns, flexibility and traceability.

5.2 MDSOA FRAMEWORK

The approach presented in this paper is based on the usage of the SOA paradigm from a model driven point of view. A meta-model has been identified for each of the three abstraction levels defined in the MDA specification [OMG 2003]: computation independent model (CIM), platform independent model (PIM) and platform specific model (PSM).

Business processes are computational independent and therefore they are related to the CIM. SOA was selected to represent the platform independent level (related to the PIM) and Web Services were chosen as the specific platform (related to the PSM) to implement SOA. In the ATHENA project [ATHENA 2004a], three specific meta-models were defined to represent each abstraction level:

- POP* (Process, Organisation, Product, + other relevant dimensions) [ATHENA 2004b] was selected as the meta-model to represent and exchange business processes. POP*, through its meta-model and UML profile, is able to represent the following enterprise dimensions: process, organisation, product, decision, and infrastructure. The POP* meta-model is a high abstraction level model (related to the CIM) representing the business aspects an organisation wants to model.

- PIM4SOA (Platform Independent Model for Service Oriented Architecture) [Benguria 2006, SourceForge 2007a] is a meta-model for representing service, process, information and quality of service elements. This meta-model allows to describe services, their collaborations, information, etc. in a platform independent (related to the PIM) way.
- WSDL (Web Service Description Language) [W3C 2001] and BPEL (Business Process Execution Language) [IBM 2003b] are some of the Web service platform specific (related to the PSM) assets.

5.3 MODEL TRANSFORMATIONS

Model transformation is one of the key MDA concepts in this approach because it allows the derivation of PIM4SOA models from POP* models in a systematic way by taking a model as input and generate another model as output. Several formal transformation languages exist in the context of MDA, like QVT [OMG 2001], MTF [IBM 2007b], ATL [AMMA 2008], MOFScript [Eclipse 2007] or UMT [SourceForge 2007b]. The transformation defines rules at meta-model level, which will later be applied at model level.

The following list documents some of the rules for the transformation between the POP* meta-model and the PIM4SOA meta-model (see Figure 5-1):

- A Process in POP* is represented as a collaboration and a task for the main collaboration in PIM4SOA
- A Role in POP* related to a Process through a “plays role” relationship, is represented as role provider in PIM4SOA
- A Flow between Processes in POP* is represented as a flow between their related tasks in PIM4SOA.
- A Product in POP* is represented as an information element in PIM4SOA

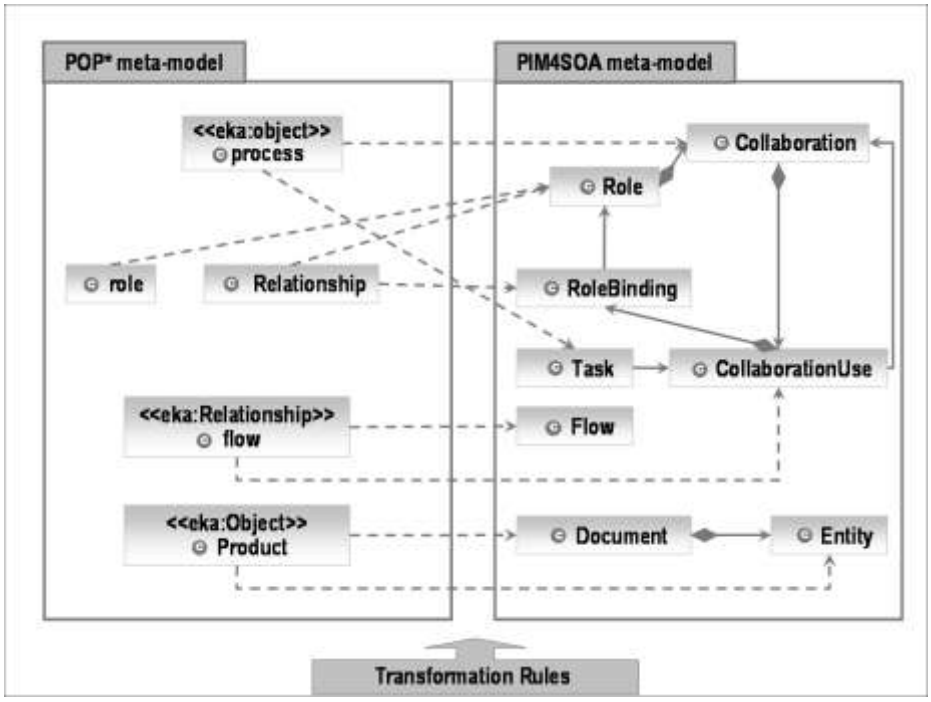

Figure 5-1: Transformation Rules

As shown in Figure 5-1, transformations do not only consist of one-to-one mappings, but also include 1-to-n and n-to-1 mappings between constructs in the POP* meta-model and those in the PIM4SOA meta-model. Although these transformations are purely syntactical conversions between two domain languages, they would benefit from being enriched with domain related semantic information for generating more accurate target models.

5.4 OPEN ISSUES AND FUTURE RESEARCH CHALLENGES

Several open issues and future research topics have been identified while carrying out a case study that was designed to prove the concept of formal and automated transformation between the CIM, PIM and PSM abstraction levels. A business process between an aircraft manufacturer and a new landing gear provider was modelled in POP* in order to apply the full chain of transformations which would convert the business model into WSDL and BPEL files that implement the supporting platform specific infrastructure.

The transformations used in the case study had finally to be implemented in Java, because existing transformation languages like MTF [IBM 2007b] or ATL [AMMA 2008] turned out to be ill-suited for addressing complex transformations which imply major structural and referential changes. A better support for handling constraint based decisions in MTF and ATL would therefore be desirable.

Furthermore, the rules applied are limited to transformations from the POP* to the PIM4SOA meta-model, which both are not yet public domain standards (although they were submitted to the pertinent standardisation bodies). It is evident that the availability of *generic* transformation rules between standardised BPM languages and domain specific PIMs would help to foster widespread usage of the MDSOA framework.

In specific cases it was furthermore not possible to correctly transform constructs from the CIM level (modelled in POP*) to constructs of the PIM (PIM4SOA) and PSM (WSDL and BPEL) levels without taking into account semantic context information which, however, was not captured in the models and thus had to be processed by human intervention. This indicates the need for semantic annotation and mediation techniques to be coupled with the different MDA abstraction layers and transformations, in order to avoid missing semantic context information that may be essential for a correct transformation of models.

In a similar sense, it would be desirable to extend the dimensions currently covered by the PIM4SOA meta-model (service, process, information, QoS), so that relevant trust, security and dependability requirements can be modelled accordingly at the PIM level.

While working on the case study it became also apparent that it is possible to identify common and recurring patterns of business processes (or parts of them), which can be mapped to similar service-oriented architecture patters. A repository of such (standard) process and architecture patterns, similar to those known from the object oriented world [Gamma 1995], would furthermore facilitate the adoption of the presented approach.

5.5 RELATED WORK

In recent years important and interesting efforts have been invested to bridge the gap between business and IT implementations through transformations [Mantell 2005, Zhao 2005, Koehler 2003, Gardner 2003, Hamadi 2003, Jiang 2003]. One of these research efforts, for instance, pro-

vides a way to transform business processes into BPEL [Gardner 2003]. This approach directly derives the platform specific assets, such as WSDL and BPEL files, from the business process at computation independent level, without applying the intermediate step via a Platform Independent Model (PIM).

5.6 CONCLUSION

In the MDSOA framework a set of transformations has been applied for converting an enterprise model into a PIM for service oriented architecture, prior to the generation of the platform specific assets. The intermediate PIM allows for a more progressive and comprehensive transformation from the business to the platform assets. However, this intermediate layer also implies more work, since additional components and transformations need to be implemented in order to obtain it. But, on the other hand, it establishes an intermediate layer that abstracts the business and the service layers from the continuous changes at the platform level; and it allows for an easy extension of the MDSOA framework towards generation of assets for different platforms (Agents, P2P, etc).

In spite of the identified (current) shortcomings and open issues concerning the MDSOA approach, it can however be stated that the work presented in this paper shows a great potential for helping to implement the well known principle of “separation of concerns”, introduced by Edsger W. Dijkstra [Dijkstra 1976], during the development of SOAs. It facilitates bridging the gap between business and system logic and thus provides the means required for flexibly and dynamically adapting the SOA according to the changes at business process level.

REFERENCES

URLs are valid as of the publication date of this document.

[AMMA 2008]
ATLAS Model Management Architecture (AMMA). *ATL: Atlas Transformation Language*. http://www.sciences.univ-nantes.fr/lina/atl/ (2008)

[ATHENA 2004a]
ATHENA. *Advanced Technologies for Interoperability of Heterogeneous Enterprise Networks and their Applications project, IST- 507849*. http://www.athena-ip.org/

[ATHENA 2004b]
ATHENA. *Report on Methodology description and guidelines definition* (Deliverable A1.3.1) http://bpmn.org/Documents/FTF/DA131-090.doc (2004)

[Ayad 2002]
Ayad, N. & Sol, H. G. "Development of New Geographically Distributed Business Models for Global Transactions." *Proceedings of the 35th Hawaiian International Conference on Systems Sciences (HICSS 02), Volume 8.* Big Island, HI (USA), January 7–10, 2002. http://csdl2.computer.org/comp/proceedings/hicss/2002/1435/08/14350230.pdf.

[Benguria 2006]
Benguria, et al. Part 1, Ch. 3, "A Platform Independent Model for Service Oriented Architectures," 23–32. *Enterprise Interoperability.* Springer, 2007.

[Dijkstra 1976]
Dijkstra, E. W. *A Discipline of Programming*. Prentice-Hall, 1976.

[Eclipse 2007]
The Eclipse Foundation. *MOFScript* . http://www.eclipse.org/gmt/mofscript/ (2007)

[Endrei 2004]
Endrei, et.al. *Patterns: service oriented architecture and web services*. http://www.redbooks.ibm.com/abstracts/sg246303.html (2004)

[Gamma 1995]
Gamma, Erich, Helm, Richard, Johnson, Ralph, & Vlissides, John. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1995.

[Gardner 2003]
Gardner, Tracy. *UML Modelling of Automated Business Processes with a Mapping to BPEL4WS*. http://www.cs.ucl.ac.uk/staff/g.piccinelli/eoows/documents/paper-gardner.pdf

[Griss 2002]
Griss, M. *Ranking IT Productivity Improvement Strategies* (Flashline, Inc. White Paper) http://martin.griss.com/pubs/WPGRISS01.pdf (2002)

[Hamadi 2003]
Hamadi, Rachid & Benatallah, Boualem. "A Petri Net-based Model for Web Service Composition," 191–200. *Proceedings of the 14th Australasian Database Conference, Volume 17.* Adelaide, South Australia, February 2003. Australian Computer Society, 2003. ISBN ~ ISSN: 1445-1336, 0-909-92595-X

[IBM 2003b]
IBM Corporation. *Business Process Execution Language (BPEL 1.1)*. ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf (2003)

[IBM 2004b]
IBM Corporation, et al. *Business Process Definition Metamodel*. http://www.bpmn.org/Documents/BPDM/OMG-BPD-2004-01-12-Revision.pdf (2004)

[IBM 2007b]
IBM Corporation. *Model Transformation Framework*. http://www.alphaworks.ibm.com/tech/mtf (2007)

[Jiang 2003]
Jiang, Ping, Mair, Q., & Newman, J. "Using UML to design distributed collaborative workflows: from UML to XPDL," 71–76. *Proceedings 12th IEEE International Workshops Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2003)*. June 9–11, 2003. ISBN: 0-7695-1963-6. http://ieeexplore.ieee.org/iel5/8713/27586/01231385.pdf

[Koehler 2003]
Koehler, Jana, Hauser, Rainer, Kapoor, Shubir, Wu, Fred Y., & Kumaran, Santhosh. "A Model-Driven Transformation Method," 186−197. *Proceedings of the 7th International Conference on Enterprise Distributed Object Computing (EDOC 2003).* Brisbane, Australia, September 16−19, 2003. IEEE Computer Society, 2003.http://portal.acm.org/toc.cfm?id=942793&type=proceeding&coll=GUIDE&dl=GUIDE&CFID=20015409&CFTOKEN=79753522

[Levitt 1983]
Levitt, T. "The Globalization of Markets." *Harvard Business Review*, May-June1983: 92−102.

[Mantell 2005]
Mantell, Keith. *From UML to BPEL-Model Driven Architecture in a Web services world.* http://www-128.ibm.com/developerworks/webservices/library/ws-uml2bpel/ (2005)

[OMG 2001]
Object Management Group. *MOF QVT Final Adopted Specification* (OMG Adopted Specification, ptc/05-11-01). http://www.omg.org/docs/ptc/05-11-01.pdf (2001)

[OMG 2003]
Object Management Group. *MDA Guide Version 1.0.1* (J. Miller and J. Mukerji, Eds.). http://www.omg.org/docs/omg/03-06-01.pdf (2003)

[Peltz 2003]
Peltz, C. "Web services Orchestration and Choreography." *Computer 36*, 10 (October 2003): 46-52. ISSN: 0018-9162.

[SourceForge 2007a]
SourceForge.NET. *PIM4SOA*. https://sourceforge.net/projects/pim4soa (2007)

[SourceForge 2007b]
SourceForge.NET. *UML model transformation tool (UMT)*. https://sourceforge.net/projects/umt-qvt/

[Troux 2002]
Troux Technologies. *UEML Unified Enterprise Modelling Language (Project, IST- 34229)*. http://www.ueml.org (2002)

[W3C 2001]
W3C. *Web Service Description Language (WSDL 1.1) (W3C Note 15 March 2001)* http://www.w3.org/TR/wsdl (2001)

[W3C 2004b]
W3C. *Web Services Architecture (W3C Working Group Note 11 February 2004)* http://www.w3.org/TR/ws-arch/ (2004)

[Zhao 2005]
Zhao, W., Bryant, B., Cao, F., Bhattacharya, K., & Hauser, R. "Transforming Business Process Models: Enabling Programming at High Level," 173–180. *Proceedings of the IEEE International Conference On Services Computing (SCC 2005)*. Orlando, FL (USA), July 11–15, 2005. http://ieeexplore.ieee.org/iel5/10249/32669/01531252.pdf?arnumber=1531252

6 Workshop Discussion Topics

PARTICIPANTS

- Sven De Labey, K. U. Leuven University, Belgium
- Nick Rossiter, Northumbria University, UK
- Daniel Schneider, Fraunhofer Institute, Germany
- Stefan Schuster, European Software Institute, Spain
- Dennis Smith, Software Engineering Institute, USA
- Xiaofei Xu, Harbin Institute of Technology, China

SUMMARY OF WORKSHOP DISCUSSION

The discussion of the workshop focused on the challenges raised in the SOA research agenda (described in the paper presented in Section 2). Each of other the papers raised a unique set of issues that contributed greater clarity to the research agenda. The discussion from the workshop is summarized below by major topic area in the research taxonomy of the SOA research agenda.

Engineering

- Components in the context of component-based software engineering have similarities to services within SOA. One major difference is that services tend to have a more business-oriented view than components.
- A significant SOA-related issue concerns the distributed, emergent behavior of systems (of systems). As a result, testing is extremely difficult—especially because of the dynamic nature of putting services together, the need for exception handling, and the fact that services can change frequently. Service specification could be based on service interfaces and corresponding basic service models. Certification can raise trust, but there will never be a 100% guarantee. For complete testing, each runtime service instance needs to be tested, which is unrealistic in many real-world settings. It may be possible to test services through simulations. There should also be a distinction between software and hardware testing in the taxonomy.
- It will be necessary for language extensions for SOA to address semantics. Java could also be extended to support model-driven architecture (MDA). One potential research area could be the exploration of the use of the Unified Modeling Language (UML) and more specifically Object Constraint Language (OCL).
- Built-in testing for services could be a viable approach with fundamental test cases that are integrated and have a designated interface for testing.
- A common understanding of semantics, as well as of how to perform semantic discovery, continues to be an open research issue. Semantic descriptions are much more than the implementation of strong, well-defined types. Issues include who defines the ontologies, what are the ripple effects of changes to an ontology, how to keep an ontology from becoming too general, and how a specific service acquires context.
- Dynamic registration and discovery of services, as well as orchestration of services at runtime, is an ongoing research issue.

- The principles of good composition of services need to be carefully researched.
- There is a strong need for concepts, methods, and techniques for migrating from legacy systems to SOA environments. One common method is to use a wrapper approach; however, this is often only a temporary solution.
- There need to be distinct analyses of reengineering for SOA environments and the evolution of existing service-oriented systems.

Business

- Because SOA involves the integration of business and technical issues, it is important for modeling approaches to be understood by business people. UML, which is the IT standard for modeling, has been identified as being difficult for business people to use. It is important to develop strategies and modeling approaches to identify significant business processes and to model them effectively.
- The use of modeling approaches, such as MDA, is growing increasingly common. However, there are not yet good approaches for transformations between different levels of abstraction, such as between Computation Independent Models (CIM), Platform Independent Models (PIM) and Platform Specific Models (PSM). As a result, implementations do not fit the intention of the interface. Tools are essential to address the problem.
- To provide for more effective modeling across levels of abstraction, several research areas are needed, including: increasing the expressiveness of transformation languages, integrating constraint expressions, and constraint-based transformation languages and patterns.
- An unsolved problem is how to identify and quantify ROI (return on investment) and SOA adoption benefits. For example, if there is a significant decrease in response time in a specific instance, what is the business value of this decrease? Significant empirical case studies are required to perform this type of analysis. ROI needs to consider not just savings in development time, but the business value such as time-to-market and agility.
- Strategy needs to be clearly separated from governance. A reference for strategy is the ATHENA project named Enterprise Interoperability Maturity Model (EIMM) [ATHENA 2008]. Strategy issues include identification of collaborators and market identification.
- Standards and processes that facilitate inter-organizational collaboration are required.
- It is important to monitor how effective services are in meeting the business needs for which they were designed.

Operations

- SLAs need to be more strongly researched. The matching of expectations and offers is a fundamental issue. The history of the use of services can then form a starting point for certifying services. SLAs are essentially a contract, and they need to be monitored in the same way that any contract is monitored. SLAs need to be in place not just for individual services, but for composite services.
- An important research area is effective ways of implementing service registries.
- Data on potential users needs to be gathered when defining services. This data can be used for developing services that are more generic and have greater flexibility.

Cross-Cutting Concerns

- Software engineers trained in object-oriented development tend to focus on small levels of granularity. For effective services, the granularity is larger; therefore, there is a need for education and training to help change this mindset.
- There is a need for governance across organizations.
- Governance needs to include stakeholder management by clearly defining who has the right to use specific services.
- A SOA adoption issue is an understanding of how a focus on business requirements and processes, as well as the need to incorporate governance, requires a mindset different from that of traditional development.

Consolidated List of References

This list is a compilation of all sources cited in this report. URLs are valid as of the publication date of this document.

[Abascal 2006]
Abascal, J., Arrue, M., Fajardo, I., & Garay, N. "An Expert-Based Usability Evaluation of the EvalAccess Web Service." *HCI-Related Papers of Interacción 2004*. Springer-Verlag, 2006.

[Afshar 2007]
Afshar M. & Moreland, B. *Keys to Successful Governance with SOA*. http://www.ebizq.net/topics/soa/features/7680.html

[Allen 2006]
Allen, P. *Service Orientation, Winning Strategies and Best Practices*. Cambridge University Press, 2006.

[AMMA 2008]
ATLAS Model Management Architecture (AMMA). *ATL: Atlas Transformation Language*. http://www.sciences.univ-nantes.fr/lina/atl/ (2008)

[Apache 2005]
The Apache Software Foundation. *Web Services - Axis*. http://ws.apache.org/axis/ (2005)

[ATHENA 2004a]
ATHENA. *Advanced Technologies for Interoperability of Heterogeneous Enterprise Networks and their Applications project, IST- 507849*. http://www.athena-ip.org/

[ATHENA 2004b]
ATHENA. *Report on Methodology description and guidelines definition* (Deliverable A1.3.1) http://bpmn.org/Documents/FTF/DA131-090.doc (2004)

[ATHENA 2008]
ATHENA. *Enterprise Interoperability Maturity Model*. http://modelbased.net/aif/solutions/singular_solutions/solution_eimm.html (2008)

[Aurrecoechea 1998]
Aurrecoechea, C., Campbell, A., & Hauw, L. "A Survey of QoS Architectures." *ACM/Springer Verlag Multimedia Systems Journal 6*, 3 (May 1998): 138–151.

[Ayad 2002]
Ayad, N. & Sol, H. G. "Development of New Geographically Distributed Business Models for Global Transactions." *Proceedings of the 35th Hawaiian International Conference on Systems Sciences (HICSS 02), Volume 8.* Big Island, HI (USA), January 7–10, 2002. http://csdl2.computer.org/comp/proceedings/hicss/2002/1435/08/14350230.pdf.

[Balzer 2004]
Balzer, Y. *Improve your SOA Project Plans*. http://www.ibm.com/developerworks/library/ws-improvesoa/ (2004)

[Benbernou 2005]
Benbernou, S., et al. "Implicit Service Calls in ActiveXML Through OWL-S," 353–365. *Proceedings of the Third International Conference on Service Oriented Computing (ICSOC05) (Lecture Notes in Computer Science 3826)*. Amsterdam, The Netherlands, December 12–15, 2005. Springer, 2005.

[Benguria 2006]
Benguria, et al. Part 1, Ch. 3, "A Platform Independent Model for Service Oriented Architectures," 23–32. *Enterprise Interoperability.* Springer, 2007.

[Bieberstein 2005]
Bieberstein, N., Bose, S., Walker, L., & Lynch, A. "Impact of Service-Oriented Architecture on Enterprise Systems, Organizational Structures, and Individuals." *IBM Systems Journal* (December 2005). http://www.highbeam.com/doc/1G1-140141564.html (2005)

[Bieberstein 2006]
Bieberstein, N., et al. *Service-Oriented Architecture Compass - Business Value, Planning and Enterprise Roadmap*. Pearson, 2006.

[Borck 2006]
Borck, J. "Planning an SOA: Gathering Around the Drawing Board." *InfoWorld* (May 2006). http://www.infoworld.com/article/06/05/08/77665_19FEsoalife2_1.html?s=feature (2006)

[Brandner 2004]
Brandner, M., Craes, M., Oellermann, F., & Zimmermann, O. "Web Services-Oriented Architecture in Production in the Finance Industry." *Informatik-Spektrum 27*, 2 (2004): 136–145.

[Brown 2006]
Brown W. & Cantor, M. *SOA Governance: How to Oversee Successful Implementation through Proven Best Practices and Methods* (IBM White Paper). ftp://ftp.software.ibm.com/software/rational/web/whitepapers/10706900_SOA_gov_model_app_v1f.pdf (2006)

[Bunse 2006]
Bunse, C., Conradi, R., Li, J., Morisio, M., Slyngstad, O.P.N., & Torchiano, M. "An Empirical Study on the Decision Making Process in Off-The-Shelf Component Based Development," 897–900. *Proceedings of the Emerging Results Track at the 28th International Conference on Software Engineering (ICSE 2006)* (Leon J. Osterweil, H., Dieter Rombach, and Mary Lou Soffa, eds.), May 2006, Shanghai, P.R. China. ACM, 2006.

[Cardelli 1999]
Cardelli, L. & Davies, R. "Service Combinators for Web Computing." *IEEE Transactions on Software Engineering 25*, 3 (1999): 309−316.

[CBDi 2006]
CBDi. "Service Level Agreements: Best Practice Report." *CBDI Journal* (December 2006). http://www.cbdiforum.com/report_summary.php3?page=/secure/interact/2006-12/service_level_agreements.php&area=silver (2006)

[Cetin 2007]
Cetin, S., Altintas, N. I., Oguztuzun, H., Dogru, A.H., Tufekci, O., & Suloglu, S. "A Mashup-Based Strategy for Migration to Service-Oriented Computing," 169–172. *Proceedings of the IEEE International Conference on Pervasive Services*, Istanbul, Turkey, July 15–20, 2007. Digital Object Identifier 10.1109/PERSER.2007.4283910 (2007)

[Chawla 2007]
Chawla, M. & Peddinti, V. "Exposing SOA Enabled C Apps as Web Services." *SOA World Magazine* (February 2007). http://webservices.sys-con.com/read/314105.htm (2007)

[Chesbrough 2006]
Chesbrough, H. & Spohrer, J. "A Research Manifesto for Services Science." *Communications of the ACM 49*, 7 (July 2006): 35–40.

[Clements 2001]
Clements, P. C. & Northrop, L. *Software Product Lines: Practices and Patterns.* Addison-Wesley, 2001.

[Constantine 2000]
Constantine, L. "What Do Users Want? Engineering Usability into Software." http://www.foruse.com/articles/whatusers.pdf (2000)

[De Labey 2006]
De Labey, S., van Dooren, M., Steegmans, E., & Service, J. *Towards Service-Oriented Programming in Java* (Technical Report CW451). http://www.cs.kuleuven.be/publicaties/reports/CW/2006/. (2006)

[Dijkstra 1976]
Dijkstra, E. W. *A Discipline of Programming*. Prentice-Hall, 1976.

[Droms 2002]
Droms, R. & Lemon, T. *The DHCP Handbook*. Sams Publishing. 2002.

[Eclipse 2007]
The Eclipse Foundation. *MOFScript* . http://www.eclipse.org/gmt/mofscript/ (2007)

[Eclipse 2008]
The Eclipse Foundation. *Web Tools Platform (WTP) Project*. http://www.eclipse.org/webtools/ (2008)

[Endrei 2004]
Endrei, et.al. *Patterns: service oriented architecture and web services*. http://www.redbooks.ibm.com/abstracts/sg246303.html (2004)

[Fitzgerald 2006]
Fitzgerald, B. & Olsson C. M. (eds.). "The Software and Services Challenge." Contribution to the Preparation of the Technology Pillar on *Software, Grids, Security and Dependability* in the European Union 7th Framework Programme. ftp://ftp.cordis.europa.eu/pub/ist/docs/directorate_d/st-ds/fp7-report_en.pdf (2006)

[Florescu 2003]
Florescu, D., Gruenhagen, A., & Kossmann, D. "XL: A Platform for Web Services." *Proceedings of the First Biennial Conf. on Innovative Data Systems Research.* Asilomar, CA (USA), January 2003. http://www.dbis.ethz.ch/research/publications/47.pdf

[Gamma 1995]
Gamma, Erich, Helm, Richard, Johnson, Ralph, & Vlissides, John. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1995.

[Gardner 2003]
Gardner, Tracy. *UML Modelling of Automated Business Processes with a Mapping to BPEL4WS*. http://www.cs.ucl.ac.uk/staff/g.piccinelli/eoows/documents/paper-gardner.pdf (2003)

[Gold-Bernstein 2006]
Gold-Bernstein, B. & So, G. *Integration and SOA: Concepts, Technologies and Best Practices*. http://www.ebizq.net/webinars/7085.html (July 26, 2006)

[Griss 2002]
Griss, M. *Ranking IT Productivity Improvement Strategies* (Flashline, Inc. White Paper) http://martin.griss.com/pubs/WPGRISS01.pdf (2002)

[Hai 2006]
Hai, L., Li, Q., & Gu, N. "Quantifying Contexts for User-Centered Web Service Discovery," 399–404. *Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC 06)*. Hong Kong, October 16–20, 2006. IEEE Computer Society, 2006.

[Hadim 2000]
Hadim, M., et al. "Service Combinators for Web Computing in Distributed Oz," Volume IV. *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2000)*. Las Vegas, NV (USA) June 24–29, 2000. CSREA Press, 2000.

[Hamadi 2003]
Hamadi, Rachid & Benatallah, Boualem. "A Petri Net-based Model for Web Service Composition," 191–200. *Proceedings of the 14th Australasian Database Conference, Volume 17.* Adelaide, South Australia, February 2003. Australian Computer Society, 2003. ISBN ~ ISSN: 1445-1336, 0-909-92595-X

[Heineman 2001]
Heineman, G. T. & Councill, W. T. *Component-Based Software Engineering: Putting the Pieces Together*. Addison-Wesley Professional, 2001.

[High 2005]
High, R., Kinder, S., & Graham, S. *IBM's SOA Foundation: An Architectural Introduction and Overview*. http://download.boulder.ibm.com/ibmdl/pub/software/dw/webservices/ws-soa-whitepaper.pdf (November 2005)

[Horn 2005]
Horn, P. "The New Discipline of Services Science." *Business Week* (January 2005). http://www.businessweek.com/technology/content/jan2005/tc20050121_8020.htm

[Houlding 2007]
Houlding, D. "From SOA to SaaS: The Driving Forces behind Today's Software Architectures." *Dr. Dobbs Portal* (May 2007). http://www.ddj.com/architect/197700752/

[IBM 2003a]
IBM Corporation. *BPEL4WS Specification v1.1*. http://www.ibm.com/developerworks (2003)

[IBM 2003b]
IBM Corporation. *Business Process Execution Language (BPEL 1.1)*. ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf (2003)

[IBM 2004a]
IBM Research. *Services Science: A New Academic Discipline?* http://www.almaden.ibm.com/asr/SSME/facsummit.pdf (2004)

[IBM 2004b]
IBM Corporation, et al. *Business Process Definition Metamodel*. http://www.bpmn.org/Documents/BPDM/OMG-BPD-2004-01-12-Revision.pdf (2004)

[IBM 2005]
IBM Business Consulting Services. *IBM Service-Oriented Modeling and Architecture*. http://www-935.ibm.com/services/us/gbs/bus/pdf/g510-5060-ibm-service-oriented-modeling-arch.pdf (2005)

[IBM 2007a]
IBM Corporation. *Web Service Level Agreements (WSLA) Project*. http://www.research.ibm.com/wsla/ (2007)

[IBM 2007b]
IBM Corporation. *Model Transformation Framework*. http://www.alphaworks.ibm.com/tech/mtf (2007)

[IBM 2008]
IBM Corporation. *Services Sciences, Management and Engineering*. https://www.research.ibm.com/ssme/ (2008)

[InfoWorld 2007]
InfoWorld. *InfoWorld Research Report: Service Oriented Architecture(SOA), April 2007.* http://www.s2.com.br/s2arquivos/403/multimidia/197Multi.pdf (2007)

[JCP 2005]
JCP. *JAX-WS 2.0 Specification*. http://www.jcp.org/en/jsr/detail?id=224 (2005)

[JCP 2006]
JCP. *Java EE 5.0 Specification*. http://www.jcp.org/en/jsr/detail?id=244 (2006)

[Jiang 2003]
Jiang, Ping, Mair, Q., & Newman, J. "Using UML to design distributed collaborative workflows: from UML to XPDL," 71–76. *Proceedings 12th IEEE International Workshops Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2003)*. June 9–11, 2003. ISBN: 0-7695-1963-6. http://ieeexplore.ieee.org/iel5/8713/27586/01231385.pdf

[Jin 2004]
Jin, J. & Nahrstedt, K. "QoS Specification Languages for Distributed Multimedia Applications: A Survey and Taxonomy." *IEEE MultiMedia, 11*, 3 (2004): 74−87.

[Jini 2007]
Jini.org. *The Community Resource for Jini Technology*. http://www.jini.org/ (2007)

[Kajko-Mattsson 2007]
Kajko-Mattsson, Mira, Lewis, Grace, & Smith, Dennis. "A Framework for Roles for Development, Evolution and Maintenance of SOA-Based Systems," 7. *Proceedings of the International Workshop on Systems Development in SOA Environments (SDSOA 2007) at the International Conference on Software Engineering (ICSE 2007)*. Minneapolis, MN (USA), May 20–26, 2007. Digital Object Identifier 10.1109/SDSOA.2007.1

[Kajko-Mattsson 2008]
Kajko-Mattsson, Mira, Lewis, Grace, & Smith, Dennis. "Evolution and Maintenance of SOA-Based Systems at SAS." *Proceedings of the 41st Hawaii International Conference on System Sciences (HICSS-41)*. Waikoloa, Big Island, Hawaii (USA), January 7–10, 2008. IEEE Computer Society, 2008. http://csdl2.computer.org/comp/proceedings/hicss/2008/3075/00/30750119.pdf

[Kempf 1999]
Kempf, J. & St. Pierre, P. *Service Location Protocol for Enterprise Networks: Implementing and Deploying a Dynamic Service Finder*. John Wiley & Sons. 1999.

[Kiczales 1997]
Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J. M., & Irwin, J. "Aspect-Oriented Programming," 220–242. *Proceedings of the 11th European Conference on Object-Oriented Programming (ECOOP '97) (Lecture Notes in Computer Science, 1241)* Jyväskylä, Finland, July 9–13, 1997. Springer, 1997.

[Kistler 1998]
Kistler, T. & Marais, H. "WebL - A Programming Language for the Web." *Computer Networks and ISDN Systems 30*, 1-7 (April 1998): 259−270.

[Koehler 2003]
Koehler, Jana, Hauser, Rainer, Kapoor, Shubir, Wu, Fred Y., & Kumaran, Santhosh. "A Model-Driven Transformation Method," 186−197. *Proceedings of the 7th International Conference on Enterprise Distributed Object Computing (EDOC 2003)*. Brisbane, Australia, September 16−19, 2003. IEEE Computer Society, 2003.http://portal.acm.org/toc.cfm?id=942793&type=proceeding&coll=GUIDE&dl=GUIDE&CFID=20015409&CFTOKEN=79753522

[Küster 2007]
Küster, U. & König-Ries, B. "Supporting Dynamics in Service Descriptions—The Key to Automatic Service Usage," 220–232. *Proceedings of Fifth International Conference on Service-Oriented Computing (ICSOC 2007)* (Lecture Notes in Computer Science, 4749). Vienna, Austria, September 17–20, 2007. Springer, 2007.

[Levitt 1983]
Levitt, T. "The Globalization of Markets." *Harvard Business Review*, May-June1983: 92−102.

[Lewis 2007]
Lewis, G., Morris, E., Simanta, S., Smith, D., & Wrage, L. "SMART: Analyzing the Reuse Potential of Legacy Components in a Service- Oriented Architecture Environment." *Proceedings of the 2007 AIAA Infotech@Aerospace Conference.* Rohnert Park, CA (USA), May, 2007. AIAA, 2007.

[Linthicum 2006]
Linthicum, D. "When Building a SOA, How Do You Know When You're Done?" *InfoWorld* (June 2006). http://weblog.infoworld.com/realworldsoa/archives/2006/06/when_building_a.html (2006)

[Manes 2007]
Manes, A. T. *SOA Governance Infrastructure*. Burton Group, 2007.

[Mantell 2005]
Mantell, Keith. *From UML to BPEL-Model Driven Architecture in a Web services world.* http://www-128.ibm.com/developerworks/webservices/library/ws-uml2bpel/ (2005)

[Marks 2006]
Marks, E., Bell, M. *Service Oriented Architecture: A Planning and Implementation Guide for Business and Technology*. John Wiley & Sons, 2006.

[Martens 2003]
Martens, A. "Usability of Web Services," 182–190. *Proceedings of the Fourth International Conference on Web Information Systems Engineering Workshops (WISEW'03)*. Rome, Italy, December 13, 2003. John Wiley & Sons, 2003.

[McClure 2006]
McClure, D. *Are IT Organizational Structures a Barrier to Business Service Management Success?* http://dougmcclure.net/blog/2006/11/are-it-organizational-structures-a-barrier-to-business-service-management-success/ (2006)

[Nielsen 1993]
Nielsen, J. *Usability Engineering*. Morgan Kaufmann, 1993

[OASIS 2007]
OASIS. *Web Services Context Specification (WS-Context) Version 1.0*. http://docs.oasis-open.org/ws-caf/ws-context/v1.0/wsctx.html (2007)

[OMG 1999]
Object Management Group. *CORBA Component Model Joint Revised Submission OMG Document orbos/99-07-01* OMG, July 1999.

[OMG 2001]
Object Management Group. *MOF QVT Final Adopted Specification* (OMG Adopted Specification, ptc/05-11-01). http://www.omg.org/docs/ptc/05-11-01.pdf (2001)

[OMG 2003]
Object Management Group. *MDA Guide Version 1.0.1* (J. Miller and J. Mukerji, Eds.). http://www.omg.org/docs/omg/03-06-01.pdf (2003)

[Pai 2007]
Pai, Y. *Ideal IT Organization for Business Agility*. http://soablueprint.com/yahoo_site_admin/assets/docs/SOA_Ideal_Organization.9153115.pdf (2007)

[Papazoglou 2003]
Papazoglou, Mike. "SOC: Concepts, Characteristics and Directions," 3–12. *Proceedings of the Fourth International Conference on Web Information Systems Engineering Workshops (WISEW'03)*. Rome, Italy, December 13, 2003. John Wiley & Sons, 2003.

[Peltz 2003]
Peltz, C. "Web services Orchestration and Choreography." *Computer 36*, 10 (October 2003): 46-52. ISSN: 0018-9162.

[Pereira 2004]
Pereira, F., Valente, M., et al. "Tactics for Remote Method Invocation." *Journal of Universal Computer Science 10*, 7 (2004): 824−842.

[Pujari 2004]
Pujari, D. "Self-Service with a Smile? Self-Service Technology (SST) Encounters among Canadian Business-to-Business." *International Journal of Service Industry Management 15*, 2 (2004): 200–219.

[Rodriguez 2005]
Rodriguez, J. *New Rules Govern SOA Lifecycle*. http://www.looselycoupled.com/opinion/2005/rodri-rules-gov0701.html (July 2005)

[Shodjai 2008]
Shodjai, P. *Web Services and the Microsoft Platform*. http://msdn2.microsoft.com/en-us/library/aa480728.aspx (2008)

[Sneed 2006]
Sneed, H. "Integrating legacy Software into a Service Oriented Architecture," 3–14. *Proceedings of the 10th European Conference on Software Maintenance (CSMR 2006)*. Bari, Italy, March 22–24 2006. IEEE Computer Society Press, 2006. http://ieeexplore.ieee.org/iel5/10671/33675/01602353.pdf?tp=&isnumber=&arnumber=1602353

[Sneed 2007]
Sneed, H. "Migrating to Web Services: A Research Framework." *Proceedings of the International Workshop on SOA Maintenance Evolution (SOAM 2007), 11th European Conference on Software Maintenance and Reengineering (CSMR 2007)*, Amsterdam, the Netherlands, March 20–23, 2007. http://www.cs.vu.nl/csmr2007/workshops/4-%20SneedSOAPaper.pdf

[Spohrer 2006]
Spohrer, J. & Maglio, P. *The Emergence of Service Science: Toward Systematic Service Innovations to Accelerate Co-creation of Value*. http://www.almaden.ibm.com/asr/SSME/jspm.pdf (2006)

[SourceForge 2007a]
SourceForge.NET. *PIM4SOA*. https://sourceforge.net/projects/pim4soa (2007)

[SourceForge 2007b]
SourceForge.NET. *UML model transformation tool (UMT)*. https://sourceforge.net/projects/umt-qvt/

[Tilley 2004]
Tilley, S., Gerdes, J., Hamilton, T., Huang, S., Müller, H. A., Smith, D., & Wong, K. "On the Business Value and Technical Challenges of Adopting Web Services." *Journal of Software Maintenance and Evolution: Research and Practice 16*, 1–2 (2004): 31–50.

[Troux 2002]
Troux Technologies. *UEML Unified Enterprise Modelling Language (Project, IST- 34229)*. http://www.ueml.org (2002)

[UDDI 2008]
UDDI XML.org. *Welcome to UDDI XML.org*. http://uddi.xml.org/ (2008)

[UPnP 2008]
UPnP Forum. *Welcome to the UPnP Forum*. http://www.upnp.org/ (2008)

[Verheecke 2004]
Verheecke, B., et al. "AOP for Dynamic Configuration and Management of Web Services." *International Journal on Web Services Research 1*, 3 (2004): 25−41.

[Veryard 2004]
Veryard, R. *The SOA LifeCycle*. CBDi, August 2004.

[W3C 2001]
W3C. *Web Services Description Language (WSDL) 1.1* (W3C Note 15 March 2001). http://www.w3.org/TR/wsdl

[W3C 2004a]
W3C. *OWL-S: Semantic Markup for Web Services*. http://www.w3.org/Submission/OWL-S/ (November 2004)

[W3C 2004b]
W3C. *Web Services Architecture (W3C Working Group Note 11 February 2004)* http://www.w3.org/TR/ws-arch/ (2004)

[W3C 2005a]
W3C. *Semantic Web Services Framework (SWSF) Overview*. http://www.w3.org/Submission/SWSF/ (September 2005)

[W3C 2005b]
W3C. *Web Service Semantics - WSDL-S*. http://www.w3.org/Submission/WSDL-S/ (November 2005)

[W3C 2007]
W3C. *Semantic Annotations for WSDL and XML Schema*. http://www.w3.org/TR/sawsdl/ (August 2007)

[Weiser 1993]
Weiser, M. "Some Computer Science Issues in Ubiquitous Computing*." Communications of the ACM 36*, 7 (1993): 75−84.

[Windley 2006]
Windley, P. "SOA Governance: Rules of the Game." *InfoWorld* (January 2006). http://www.infoworld.com/pdf/special_report/2006/04SRsoagov.pdf

[Winter 2007]
Winter, A. & Ziemann, J. "Model-Based Migration to Service-Oriented Architectures." *Proceedings of the International Workshop on SOA Maintenance Evolution (SOAM 2007), 11th European Conference on Software Maintenance and Reengineering (CSMR 2007)*, Amsterdam, the Netherlands, March 20–23, 2007. http://www.cs.vu.nl/csmr2007/workshops/2-%20winterziemann.pdf

[Woolf 2005]
Woolf, B. *Streamline SOA Development Using Service Mocks*. http://www-128.ibm.com/developerworks/library/ws-mocks/ (2005)

[WSMO 2008]
WSMO. *Web Service Modeling Ontology*. http://www.wsmo.org (2008)

[X500 2008]
X500Standard.com. *Welcome*. http://www.x500standard.com/ (2008)

[Zhao 2005]
Zhao, W., Bryant, B., Cao, F., Bhattacharya, K., & Hauser, R. "Transforming Business Process Models: Enabling Programming at High Level," 173–180. *Proceedings of the IEEE International Conference On Services Computing (SCC 2005)*. Orlando, FL (USA), July 11–15, 2005. http://ieeexplore.ieee.org/iel5/10249/32669/01531252.pdf?arnumber=1531252

[Ziemann 2006]
Ziemann, J., Leyking, K., Kahl, T., & Werth, D. "SOA Development Based on Enterprise Models and Existing IT Systems." *Exploiting the Knowledge Economy: Issues, Applications and Case Studies* (P. Cunningham & M. Cunningham, eds.). IOS Press, 2006.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	July 2008	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Proceedings of the International Workshop on the Foundations of Service-Oriented Architecture (FSOA 2007)	FA8721-05-C-0003

6. AUTHOR(S)
Grace A. Lewis and Dennis B. Smith

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2008-SR-011

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

This report presents the results of the Foundations of Software-Oriented Architecture (FSOA) workshop held at the Third International Conference on Interoperability for Enterprise Software and Applications (I-ESA 2007). This workshop was organized to provide a forum for a concerted effort to develop a long-term, community-wide research agenda to bridge the gap between SOA research and the real needs of the practitioners in the field. An initial research agenda for SOA was presented along with three papers that focus on specific aspects of operations, engineering, and business challenges. The papers are each presented in this report, and the discussion and its implications for an evolving research agenda are summarized.

14. SUBJECT TERMS	15. NUMBER OF PAGES
SOA, service-oriented architecture	84

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL